F 9/9/15

6.02 Lecture #1

Fernando Trujano

mit.edu/6.02 Digital Communications Systems

Digital : message comprising a discrete-time sequence of symbols from source alphabet

Analog : continious-time waveform

Measure and Information:

$$I_\alpha = \log_2 \frac{1}{P_\alpha}$$

Expected information:

Entropy and source: $H = \sum_\alpha P_\alpha I_\alpha = \sum_\alpha P_\alpha \log_2 \frac{1}{P_\alpha}$

9/10/15

6.02 Recitation

Fernando Troja

Probabilistic Model

event has occured if outcome is in the event set

Space and outcomes

Ex: V = vowel = $\{a, e, i, o, u\}$

F = First five = $\{a, b, c, d, e\}$

$P(V) = P(a) + P(e) + P(i) + P(o) + P(u)$

$P(F) = P(a) + P(b) + P(c) + P(d) + P(e)$

$P(V \text{ or } F) = P(V) + P(F) - P(V \text{ and } F)$

because $P(V)$ and $P(F)$ are not mutually exclusive

Conditional Probability

$$P(V \mid F) = \frac{P(V \text{ and } F)}{P(F)}$$

↑ given that

IF $P(V \mid F) = P(V)$

$\Rightarrow$ V and F are independent

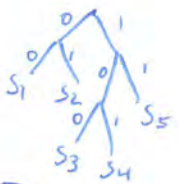
$\Rightarrow$ $P(V \text{ and } F) = P(V) * P(F)$

$P(V \text{ and } F) = P(V \mid F)\, P(F)$

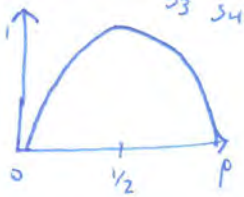
Baye's Rule $= P(F \mid V)\, P(V)$ by symmetry

$$\Rightarrow P(V \mid F) = \frac{P(F \mid V)\, P(V)}{P(F)}$$

Prefix free: All symbols are at the leaves of the tree



Entropy: $H = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} = h(p)$



9/14/15

6.02 Lecture #2

Fernando Trujano

Prefix-free: All symbols at the leaves of the tree.

each source symbol s_i chosen independently with probability p_i
each s_i is drawn from the same probability distribution.

$L = \sum p_i l_i$ $\quad L \geq H$ ← lower bound

Efficiency: Encode information / symbols using the least amount of bits.
↖ depends on prob of symbol occuring

Claim: exists a code where $H \leq L < H+1$
Proof in notes!

9/15/15

6.02 Recitation

Fernando Troja

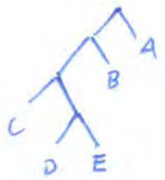
Huffman Encoding

- variable length
- shorter codeword → more likely

Algorithm

- sort based on p_i
- merge two least probable (combine probs)
- repeat until done

Sybol	D
A	.4
B	.2
C	.2
D	.1
E	.1



$L = 1(.4) + 2(.2) + 3(.2) + 4(.1) + 4(.1) =$

LZW Encoding

- need not know stats
- sender and receiver store what has been sent => library

6.02 Notes

Lecture 2: LZW and Huffman

Prefix free - all symbols on leaves of code trees

$L \geq H$ (entropy)

Huffman Encoding

- create code with minimum expected codeword length
  1) Sort list by probability
  2) Combine 2 symbols with least probability
- Good if probabilities are known ahead of time

Lempel-Ziv-Welch (LZW) Encoding/Compression

- No need to know probabilities
- Dictionary maps indeces to variable-length strings
  - initialized with all symbols

6.02 Notes

Lecture 3: Binary Symmetric Channel
Hamming Distance, Repetition Codes

source → source coding → channel coding

Adapt to characteristics of source
Huffman, LZW

Adapt to characteristics of channel

Binary Symmetric Channel - Abstraction, model errors
bits flipped independently with $p < .5$

coding rate - # useful message bits / # total bits sent

Hamming Distance (HD) = distance b/w codewords of equal length

Easy way to calculate => Add all bits mod 2 with no carrying (XOR)

So:
```
  01101000 11
+ 10010010 11
------------
  11111010 00
```
0 + 1 = 1
1 + 1 = 0
0 + 0 = 0

HD = # 1's in addition = weight of string

Minimum HD = minimum HD b/w any two valid codewords = d

Detect t errors if $d \geq t+1$
Correct t errors if $d \geq 2t+1$

can detect all patterns up to t_D bit errors while simultaneously correcting all patterns up to t_C bit errors ($t_C \leq t_D$) iff $d >= t_C + t_D + 1$

Parity Check

Add bit to end of message to make # 1's even => ↑ HD

d = 2 => detect 1 bit error, correct none

Block Coding (use parity check idea)

Add n-k bits to message to "protect" as many bits as possible

rate = k/n

1/1

6.02 Notes

Lecture 4: Linear Block Codes

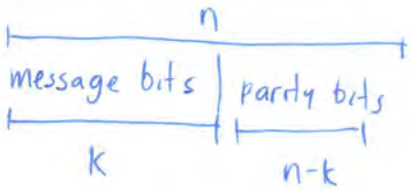
Block Codes

message $\Rightarrow$ k bit blocks $\rightarrow$ n bit codeword $\xrightarrow{\text{send}}$

rate $= k/n$ $\Rightarrow (n, k)$ codes or (n, k, d) ← min HD

2^n code words to chose from : 2^k to select

systematic codes:



Linear block Codes

- Sum of two codewords also a codeword
  ↑ sum mod 2 aka XOR

minimum HD = smallest weight among non-zero codewords

Generator Matrix G
← data k elm row

$c = d\,G$

↑ codeword, n elm row; ↖ k x n Generator matrix

we want the first k bits of c to = first k bits of d
↳ since systematic codes (adding parity to end)

$\Rightarrow$ $G = [I \mid A]$

Rectangular Codes

- calculate parity bit of each row and column

Rate $= rc/(rc + r + c)$ (r ← rows, c ← columns)

d_1	d_2	p_1
d_3	d_4	p_2
p_3	p_4	

In systematic form we have $[d_1\ d_2\ d_3\ d_4 \mid p_1\ p_2\ p_3\ p_4]$

so (I since systematic)

$$G = \left[\begin{array}{cccc:cccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right] \begin{matrix} d_4 \\ d_2 \\ d_3 \\ d_4 \end{matrix}$$

First parity bit covers 1st 2nd data bits

p_4 coverd d_2 and d_4

$\Rightarrow$ SEC = Single Error Correction

could be more efficient $\Rightarrow$ Hamming Codes

Hamming Codes

correct single bit errors with min # parity bits

$$n = 2^{n-k} - 1$$

$P_1 = d_1 + d_2 + d_3$
$P_2 = d_1 + d_3 + d_4$
$P_3 = d_2 + d_3 + d_4$

more on next lecture...

Bit positions with powers of 2 = parity bit
the rest is data bits

=> P_1 protects all bits that have least sig dig set
P_2 " " " second " "
P_3 " " " third "

6.02 Notes

Lecture 5: Syndrome Decoding

Syndrome Decoding

- efficient way to decode any linear block code

Recall: encoding using G where $dG = c$ $G = [I \mid A]$

For decoding $Hr^T = s$ ← syndrome bits

↑ received codeword

$H = [A^T \mid I]$

$He^T = s$

↑ error vector (which bits to flip)

1) Syndrome table:

FOR SEC populate with He^T for each e

error vectors

1000..
0100..
0010..
0001..
....

2) For each received codeword find Hr^T

6.02 Notes

Chapter 5: Coping with bit errors using error correction codes

BSC = Binary symetric channel => P of bit flip independent

Repetition Code

010100 → 00 11 00 . . . .

n n n

Code rate = 1/n

↑ For every usefull messaye bit, we transmit n

Can find and correct error if pattern is off with most likely pattern.

0001000 → 00000

Hamming Distance

of bit positions in which messages m_1 and m_2 differ

IF HD = 1 => can't tell single bit errors appart

Error Detection: HD - 1 errors

Error Correction: $\left\lfloor \frac{HD-1}{2} \right\rfloor$ errors

Linear Block Codes

(n, k) codes => k message bits produce n code bits $n \geq k$

rate = k/n

Arithmetic modulo 2:

0+0 = 0 0+1 = 1 like XOR

1+1 = 0 1+0 = 1

Thm: A code is linear iff the sum of any two codewords is another codew

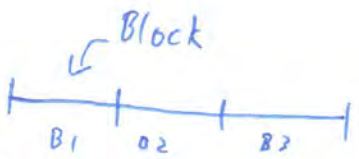
MHD = weight of code with least ↑non-0 weight, # of 1's

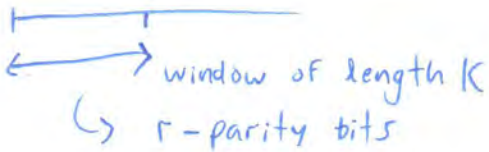
11/29/15

6.02 Recitation

Convolution Code

- linear code → compute parity only
- block vs stream.
- sliding window, length K and r parity bits
- error correction, each message bit affects many parity bits





Parity Equations:

$$P_0[n] = x[n] + x[n-1] + x[n-2]$$
$$P_1[n] = x[n] + x[n-1]$$
$$P_2[n] = x[n] + x[n-2]$$

window, $K = 3$
parity bits $r = 3$
rate $= 1/r = 1/3$

view each parity equation ($P_i[n]$) w/ a generator polynomial $g_i[n]$

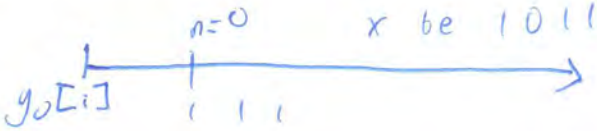
$g_0[n]$ are $(1, 1, 1)$
$g_1[n]$ $(1, 1, 0)$
$g_2[n]$ $(1, 0, 1)$

$\begin{pmatrix} x[n] \\ x[n-1] \\ x[n-2] \end{pmatrix}$ ← what's actually happening

$$P_i[n] = \sum_{j=0}^{k-1} g_i[j]\, x[n-j]$$

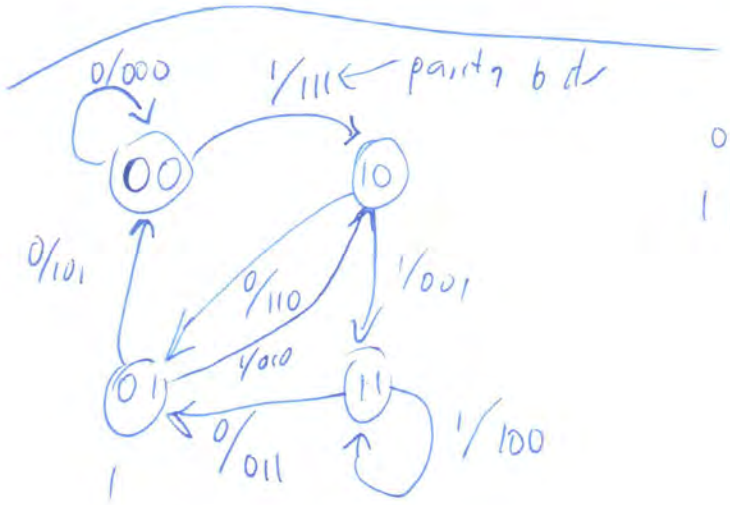
↑ location of window

← Flip in time



1 1 0 1 0 0 0 0 $x[-j]$ $n=0$
1 1 0 1 0 0 0 $x[1-j]$ $n=1$
1 1 0 1 0 0 $x[2-j]$ $n=2$
1 1 0 1 0 $x[3-j]$ $n=3$

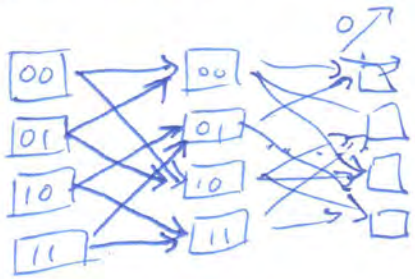
	$V_1 =$	0	1	2	3	4	5
XOR	$P_0 =$	1	1	0	0	0	1



0	00	→ 00
1	00	→ 10

new states = next piece of data

Alternative view:



6.02 Notes

Lecture 6: Convolutional Coding

Convolutional Codes

- Very efficient
- work on continuous stream of bits
  - vs linear breaks bitstream into blocks
- transmit non-systematic codes
  - ↳ only parity bit

Example code

$p_0 = x[n] + x[n-1] + x[n-2]$

$p_1 = x[n] + x[n-2]$

Message: 11001101

(pad with zeroes at beginning)

00110011 01

$n=0$

$x[n]=1 \quad x[n-1]=0 \quad x[n-2]=0 \Rightarrow p_0[0]=1, \; p_1[0]=1$

00110011 01

$x[n-2]$ $x[n-1]$ $x[n]$

$p_0[1]=0, \; p_1[1]=1$

$p_0[2]=0, \; p_1[2]=1$

Transmit: 11 01 01 ...

only parity bits!

Looking at 3 bits a time

using a sliding window of width = 3 $\Rightarrow$ $K=3$ ← code's constraint length

Each message bit protected by K parity bits

The greater K is => more redundance => ↑ error correction possibilities

Generator: $G_0 = 1\,1\,1$, $G_1 = 1\,0\,1$ — 1 in j^{th} position if x

location of window

$$p_i[n] = \sum_{j=0}^{K-1} g_i[j]\, x[n-j]$$

input bitstream as a function of j reversed in time

6.02 Notes

Lecture 7: Viterbi decoding

Encoding: Create path through trellis, sent parity bits

Decoding: Find most likely path that transmitter took

Viterbi Decoding

Branch Metric (BM) - HD b/w received parity bits and parity bits of transitions

↳ How likely it is that we did NOT make that transition

* At each timestep find most likely path to each state at that time

Path Metric P[s,i] - smallest sum of branch metrics, minimized over all message sequences that place the transmitter in state s at time i

Find PM[s,i+1] using PM[s,i] and BM for the incoming branches

Algorithm:

For each state s at time i+1

```
a,b = predecessor_states(s) //states transmitter could've been at time i
// calculate branch metrics for each
BMa = ~
BMb = ~

PM[s,i] = min( PM[a,i] + BMa , PM[b,i] + BMb)
// Keep track of predecessor state used to get to s in iteration i
```

6.02 Recitation

Quiz I Review

$$H = E = \sum p_i \log_2\left(\frac{1}{p_i}\right)$$

LBC

parity bits

$2^{n-k} \geq n + 1$ ← case where no error

Single bit errors

possible configurations of parity bits

2010 F QI P2

~~$2^{n-k} \geq \binom{n}{1}$~~

Correct for t errors

$$2^{n-k} \geq 1 + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{t}$$

does a $(11, 5, 5)$ (n, k, d) work

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor = \frac{4}{2} = 2$$

need 2 t errors

Verify that $2^{11-5} \geq 1 + \binom{11}{1} + \binom{11}{2}$?

Convolutional Codes F2011 Q1

6.02 Notes

Chapter 9 Noise

Additive white Gaussian Noise (AWGN)

$$y[k] = y_0[k] + w[k]$$

$y[k]$: received; $y_0[k]$: sent; $w[k]$: Noise ~ Gaussian w/ mean = 0

PDF - Probability Density Function $f_w(w)$

$$P(w_0 \leq W \leq w_0 + dw) \approx f_w(w_0)\, dw$$

$$P(w_1 \leq W \leq w_2) = \int_{w_1}^{w_2} f_w(\alpha)\, d\alpha$$

Note that $\int_{-\infty}^{\infty} f_w(\alpha)\, d\alpha = 1$

mean:

$$\mu_w = \int_{-\infty}^{\infty} \alpha f_w(\alpha)\, d\alpha$$

$$\sigma_w^2 = \int_{-\infty}^{\infty} (\alpha - \mu_w)^2 f_w(\alpha)\, d\alpha$$

$Q(w)$: Probability of a standard Gaussian random variable taking values greater than w

$Q\left(\frac{w-\mu}{\sigma}\right)$: Probability of taking values greater than w for Gaussian distribution

Distance of w from mean μ, measured in standard deviations

Bit Error Rate (BER): probability that any given bit is in error

$$BER = Q\left(\frac{V_1 - V_0}{2\sigma}\right)$$ - Equal Prior Probabilities

$$BER_{on\text{-}off} = Q\left(\frac{V_e}{2\sigma}\right)$$ 0 or V

$$BER_{bipolar} = Q\left(\frac{V_e}{\sigma}\right)$$ -V or V

SNR - Signal to Noise Ratio

- ratio of signal power to noise variance

- Scaling wouldn't work as it would also scale the noise

Solution: Average k samples in the bit slot

↳ mean still = 0 but var = σ^2/k instead of σ^2

=> SNR ↑ by Factor of k

6.02 Notes

Chapter 10

LTI Systems ← Linear Time Invariant

$x[n] \rightarrow [S] \rightarrow y[n]$ ← output

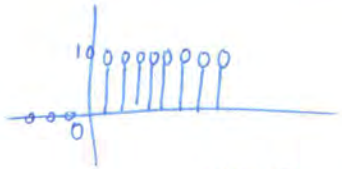
↑ input

Time invariant → $x[n-D] \rightarrow [S] \rightarrow y[n-D]$

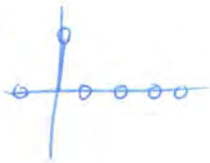
↑ no need to know n, can just look at time difference

Casual Models: $y[n]$ only depends on past and present values of $x[\cdot]$

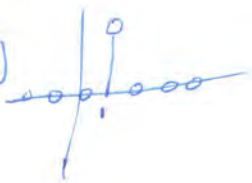
Unit Step: $u[n]$



Unit Sample $\delta[n]$



$\delta[n-1]$



$\delta[n] = u[n] - u[n-1]$

$u[n] \rightarrow [S] \rightarrow s[n]$

↑ unit step response

$\delta[n] \rightarrow [S] \rightarrow h[n]$

↑ unit sample response

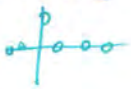
6.02 Notes

Chapter 11

Can characterize an LTI system by knowing $h[n]$ ← response to $\delta[n]$

- can write signal $x[n]$ as sum of scaled and shifted unit sample functions $\delta[n]$

$$x[n] = \ldots + x[-1]\delta[n+1] + x[0]\delta[n] + \ldots + x[k]\delta[n-k] + \ldots$$



$$\Rightarrow \quad x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \rightarrow \boxed{S} \rightarrow y[n] = \sum_{k=-\infty}^{\infty} x[k]\,h[n-k]$$

since $\delta[n] \rightarrow \boxed{S} \rightarrow h[n]$

CONVOLUTION

$$y[n] = x * h = h * x$$

convolved with

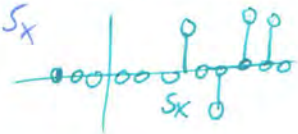
convolution is communative

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]\,x[n-m]$$

change of vars

With casual systems input is zero at begenning before a start time S_x

$$\Rightarrow y[n] = \sum_{k=S_x}^{n} x[k]\,y[n-k]$$



Convolution is also acossiative:

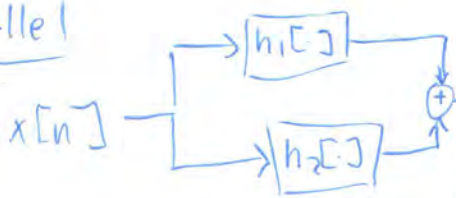
$$(h_2 * h_1) * x = h_2 * (h_1 * x)$$

Series:

$x[n] \rightarrow \boxed{h_1} \rightarrow \boxed{h_2} \rightarrow y[n] \quad = \quad x[n] \rightarrow \boxed{h_2 * h_1[\cdot]} \rightarrow y[n]$

$= x[n] \rightarrow \boxed{h_1 * h_2[\cdot]} \rightarrow y[n] \quad = \quad x[n] \rightarrow \boxed{h_2} \rightarrow \boxed{h_1} \rightarrow y[n]$

Parallel



$x[n] \rightarrow \boxed{(h_1+h_2)[\cdot]} \rightarrow y[n]$

$y = y_1 + y_2 = (h_1 * x) + (h_2 * x) = (h_1 + h_2) * x$

Flip · Slide · Dot

$$y[n_0] = \sum_{k=-\infty}^{\infty} x[k]\,h[n_0 - k]$$

$h[n_0 - k] = h[-k + n_0]$ = h reversed in time $h[-k]$ shifted n_0 to the right $+n_0$

Deconvolution

Process $y[n]$ through LTI Filter designed to cancel effect of channel

$x[n] \rightarrow \boxed{h_1[\cdot]} \rightarrow y[n] \rightarrow \boxed{h_2[\cdot]} \rightarrow z[n]$ where $x[n] = z[n]$

To do that we ~~not~~ need to satisfy:

$$(h_2 * h_1)[n] = \delta[n]$$

since $z = h_2 * (h_1 * x) = (h_2 * h_1) * x = \delta_0 * x = x$

fun fact: convolving any signal w/ unit sample yields that signal back

Find h_2 using Flip-Slide-Dot

$s[n]$ and $h[n]$

b/c of superposition and time invariance

Since $s[n] = u[n] - u[n-1] \Rightarrow h[n] = s[n] - s[n-1]$

So if casual system and $h[k] = 0$ and $s[k] = 0$

$$s[n] = \sum_{0}^{n} h[k]$$

no need for convolution woooooo

Eye Diagrams

TODO

12

6.02 Notes

Chapter 12: Frequency Response of LTI Systems

Periodic if $x[n+P] = x[n]$ (Fixed positive integer)

smallest P = period

Periodic inputs with frequency Λ → periodic outputs with frequency Ω

Discrete Time Sinusoid

$$x[n] = \cos(\Omega_0 n + \theta_0)$$ (Ω_0: angular frequency)

* All action of interest happens at the $[-\pi, \pi]$ range

$\Omega_0 = 0 \Rightarrow$ lowest variation

$\Omega_0 = \pm\pi \Rightarrow (-1)^n \Rightarrow$ highest variation

Complex Exponentials

$$e^{j\phi} = \cos\phi + j\sin\phi \quad \leftarrow \text{Euler}$$

($j = \sqrt{-1}$; Real Component: $\cos\phi$; Immaginary Component: $j\sin\phi$)

Recall: $c = a + jb$

$|c| = \sqrt{a^2+b^2}$

$\text{angle}(c) = \arctan(b/a)$

$a = |c|\cos(\angle c)$

$b = |c|\sin(\angle c)$

Polar form: $c = |c|e^{j\angle c}$

Point on the unit circle in the complex plane (radius = 1) and at an angle of ϕ to positive real axis

More Identities

$$\cos\phi = \frac{1}{2}\left(e^{j\phi} + e^{-j\phi}\right)$$

$$\sin\phi = \frac{1}{2j}\left(e^{j\phi} - e^{-j\phi}\right) = \frac{j}{2}\left(e^{-j\phi} - e^{j\phi}\right)$$

$$\cos(A)\cos(B) = \frac{1}{2}\left(\cos(A+B) + \cos(A-B)\right)$$

$$\cos(A \pm B) = \cos(A)\cos(B) \mp \sin(A)\sin(B)$$

Frequency Response

Derivation

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]\, x[n-m]$$ From last chapter

$$= \sum_{m=-\infty}^{\infty} h[m] \cos(\Omega_0(n-m) + \theta_0)$$ substitution

$$= \underbrace{\left(\sum_{-\infty}^{\infty} h[m]\cos(\Omega_0 m)\right)}_{C(\Omega)} \cos(\Omega_0 n + \theta_0) + \underbrace{\left(\sum_{-\infty}^{\infty} h[m]\sin(\Omega_0 m)\right)}_{S(\Omega)} \sin(\Omega_0 n + \theta_0)$$ Identity

$$= C(\Omega_0)\cos(\Omega_0 n + \theta_0) + S(\Omega_0)\sin(\Omega_0 n + \theta_0)$$

$$H(\Omega) = C(\Omega) - jS(\Omega) = \ldots +$$

↑ Frequency Response

$$y[n] = |H(\Omega_0)| \cos(\Omega_0 n + \theta_0 + \angle H(\Omega_0))$$

* can get $y[n]$ using only Frequency!

Say $x[n] = A_0 e^{j(\Omega_0 n + \theta_0)}$

$$y[n] = \sum_{-\infty}^{\infty} h[m]\, x[n-m]$$ Last chapter

$$= \sum_{-\infty}^{\infty} h[m]\, A_0 e^{j(\Omega_0(n-m) + \theta_0)}$$ Substitution

$$= \underbrace{\left(\sum_{-\infty}^{\infty} h[m]\, e^{-j\Omega_0 m}\right)}_{C(\Omega) - jS(\Omega) = H(\Omega)} \underbrace{A_0 e^{j(\Omega_0 n + \theta_0)}}_{x[n]}$$ Math

$\Rightarrow$ If input $x[n]$ is a complex exponential $\rightarrow$ $y[n] = H(\Omega)e^{j\Omega n}$

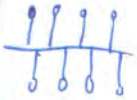
(as $e^{j\Omega n}$)

Properties of Frequency Response

If $h_D[n] = h[n-D]$ then $H_D(\Omega) = e^{-j\Omega D} H(\Omega)$ and $|H_D(\Omega)| = |H(\Omega)|$ $\Rightarrow$ Magnitude unchanged, Phase modified

Periodicity in Ω: $H(\Omega)$ repeats periodically on Ω axis with period 2π
$\Rightarrow$ Area of interest: $|\Omega| \leq \pi$

Lowest Frequency: $\boxed{H(0) = \sum_{n=-\infty}^{\infty} h[n]}$ so, if $\Omega = 0$ we get constant DC input.
← DC gain of system

Highest Frequency: $\boxed{H(\pm\pi) = \sum_{n=-\infty}^{n=\infty} (-1)^n h[n]}$ 

Symmetry: For real $h[\cdot]$ $C(\Omega)$ (← real part) is an even function of frequency $\Rightarrow C(-\Omega) = C(\Omega)$
$S(\Omega)$ " " "
$\Rightarrow |H(\Omega)|$ is even and $\angle H(\Omega)$ is odd

Real and Even: If $h[\cdot]$ is even in time $\Rightarrow$ Frequency response is purely real

Real and Odd: If $h[\cdot]$ is odd in time $\Rightarrow$ " " immaginary

$H(\Omega)$ in Series: $H(\Omega) = H_2(\Omega)H_1(\Omega) = H_1(\Omega)H_2(\Omega)$
$h[n] = (h_1 * h_2)[n] \Rightarrow H(\Omega) = H_1(\Omega)H_2(\Omega)$
So, convolution in time $\Rightarrow$ multiplication in Frequency

$H(\Omega)$ in Parallel: $h[n] = (h_1 + h_2)[n] \Rightarrow H(\Omega) = H_1(\Omega) + H_2(\Omega)$

$H(\Omega) \Rightarrow h[n]$

We know $H(\Omega) = \sum_{n=-\infty}^{\infty} h[m] e^{-j\Omega m}$

$h[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(\Omega) e^{j\Omega n} d\Omega$ ← Magical Math magic

or

$h[n] = \frac{1}{2\pi} \int_{\langle 2\pi \rangle} H(\Omega) e^{j\Omega n} d\Omega$ ← any contiguous interval of length 2π

So, $h[n]$ is represented as a weighted combination of a continuum of exponentials of the form $e^{j\Omega n}$ w/ frequencies Ω in a 2π range and associative weights $H(\Omega)d\Omega$

10/27/15

6.02 Recitation

Review:

$$x[n] \rightarrow \boxed{h[\cdot]} \rightarrow y[n] = \sum_k x[k]\,h[n-k]$$

$$x[n] = e^{j\Omega_0 n} \rightarrow \boxed{h[\cdot]} \rightarrow y[n] = H(\Omega_0)\,x[n] \quad \text{where} \quad H(\Omega) = \sum_m h[m]\,e^{-j\Omega m}$$

$x[n]$ is a sum of complex exponentials

$$x[n] \rightarrow \boxed{h[\,]} \rightarrow y[n]$$

$$X(\Omega) \quad H(\Omega) \quad Y(\Omega) \Leftarrow \text{DTFT}$$

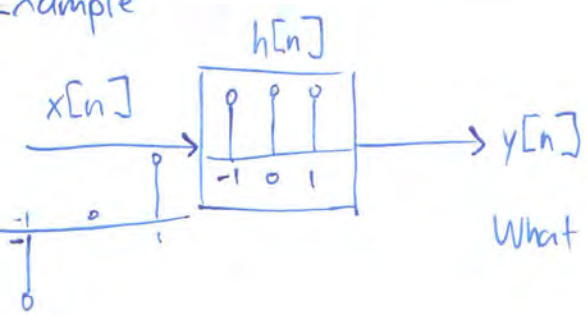
$$X(\Omega) = \sum_m x[m]\,e^{-j\Omega m}$$

$$x[n] = \frac{1}{2\pi} \int_{\langle 2\pi \rangle} X(\Omega)\,e^{j\Omega n}\,d\Omega$$

- Everything about $h[n] \longleftrightarrow H(\Omega)$ ✓
- DTFT = another way to determine $y[n]$

$$Y(\Omega) = H(\Omega)X(\Omega) \Rightarrow y[n]$$

Example



$x[n] = -\delta[n+1] + \delta[n-1]$

$h[n] = \delta[n+1] + \delta[n] + \delta[n-1]$

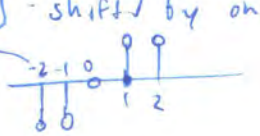
What is $y[n]$?

One way to do it: Convolution

1) Flip $x[n]$: $x[-n]$ =

2) Calculate $x[n] * h[n]$ - shift by one and see where they align

$h[k]$ =



$= x[n] * h[n] \Rightarrow y[n] = -\delta[n+2] + \delta[n+1] + \delta[n-1] + \delta[n-2]$

Can also do it:

$$h[n] = \delta[n+1] + \delta[n] + \delta[n-1]$$

1) Calculate Frequency Responce

$$H(\Omega) = \sum_m h[m] e^{-j\Omega m}$$

$$H(\Omega) = \underset{m=-1}{e^{-j\Omega}} + \underset{m=0}{1} + \underset{m=1}{e^{j\Omega}}$$

$$= 2\cos\Omega + 1 \qquad \text{because of Euler}$$

$$\sin\theta = \frac{-e^{-j\theta} + e^{j\theta}}{2j}$$

$$x[n] = -\delta[n+1] + \delta[n-1]$$

$$X(\Omega) = -e^{-j\Omega} + e^{j\Omega}$$

$$= -2j\sin\Omega$$

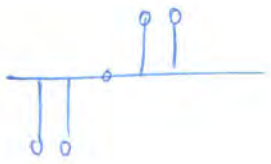
2) $Y(\Omega) = H(\Omega)\,X(\Omega)$

$$= (2\cos\Omega + 1)(-2j\sin\Omega)$$

$$= -4j\cos\Omega\sin\Omega - 2j\sin\Omega$$

$$= -2j\sin 2\Omega - 2j\sin\Omega \qquad \text{double angle Formula}$$

$$Y(\Omega) = -e^{-j2\Omega} + e^{j2\Omega} + \left(-e^{-j\Omega} + e^{j\Omega}\right) \qquad \text{Euler again}$$

So $y[n] =$  $\Rightarrow$ Same thing we got with convolution

DTFT Properties:

$$y[n] = x[n-n_0]$$

$$\Rightarrow Y(\Omega) = \sum_n x[n-n_0] \cdot e^{-j\Omega n}$$

let $m = n - n_0$ $\Rightarrow$ change of variables

$$= \sum_m x[m] e^{-j\Omega(m+n_0)}$$

$$= e^{-j\Omega n_0} X(\Omega)$$

$\Rightarrow$ Shift in Time = Linear phase addition in Frequency

Initial Value:

$$X(0) = \sum_n x[n]$$

If $x[n]$ is real

$$X(\Omega) = \sum_n x[n] e^{-j\Omega n} = \sum_n x[n]\left[\cos(-\Omega n) + j\sin(-\Omega n)\right]$$

($X(\Omega)$: real part $X(\Omega)$ | imaginary)

we know

$\mathrm{Re}(X(\Omega))$ (real part) is even and $\mathrm{Im}(X(\Omega))$ (imaginary) is odd

$\Rightarrow |X(\Omega)|$ (magnitude) is even and $\angle X(\Omega)$ (phase) is odd

$x[n]$ is real x_{even}

$$x_{even} = \frac{1}{2}\left[x[n] + x[-n]\right]$$ (note it is even)

$$X(\Omega) = \frac{1}{2}\left(\sum_n x[n] e^{-j\Omega n} + \sum_n x[-n] e^{-j\Omega n}\right)$$ change of variable

$$= \frac{1}{2}\left(\sum_n x[n] e^{-j\Omega n} + \sum_m x[m] e^{+j\Omega m}\right)$$ let $m = -n$

$$= \mathrm{Re}(X(\Omega)) + j\,\mathrm{Im}(X(\Omega))$$
$$+ \mathrm{Re}(X(\Omega)) - j\,\mathrm{Im}(X(\Omega))$$

$\Rightarrow$

$X(\Omega)$ is real

Similarly ...

$X_{odd} \Rightarrow j\,\mathrm{Im}(X(\Omega))$ (all imaginary)

6.02 Notes

Chapter 13
Fourier analysis and spectral representation of signals

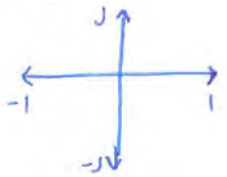
Fourier analysis - decomposing signal into weighted sum of complex exponentials

DTFT of $x[\cdot] = X(\Omega) = \sum_{m=-\infty}^{\infty} x[m]\, e^{-j\Omega m}$ = spectrum or spectral distribution or spectral content

SKIP

DTFS of periodic $x[\cdot]$

$$x[n] = \sum_{k=\langle P \rangle} A_k e^{j\Omega_k n}$$



$\Omega_k = k\Omega_1$ $\quad \Omega_0 = 0$

$\Omega_1 = \frac{2\pi}{P}$

↑ Fundamental frequency

Ω_k = k^{th} harmonic

$A_k = A_{-k}^*$

$A_{P/2} = A_{-P/2}$ = all real

DTFS Analysis Equation: $A_k = \frac{1}{P} \sum_{n=\langle P \rangle} x[n]\, e^{-j\Omega_k n}$

$$A_0 = \frac{1}{P} \sum_{n=\langle P \rangle} x[n]$$

$$A_{P/2} = A_{-P/2} = \frac{1}{P} \sum_{n=\langle P \rangle} (-1)^n x[n]$$

Even P:

$$x[n] = A_0 + \sum_{k=1}^{P/2} 2|A_k| \cos(\Omega_k n + \angle A_k)$$

Odd P

$$x[n] = A_0 + \sum_{k=1}^{(P-1)/2} 2|A_k| \cos(\Omega_k n + \angle A_k)$$

$$y[n] = \frac{1}{P} \sum_{k=\langle P \rangle} H(\Omega_k) X_k e^{j\Omega_k n} \quad \Rightarrow Y_k = H(\Omega_k) X_k$$

6.02 Notes

Chapter 14
Modulation and Demodulation

Modulation

$$X(\Omega) = \sum x[m]\, e^{-j\Omega m}$$

modulated signal → $r[n] = x[n]\cos(\Omega_c n)$

$$R(\Omega) = \frac{1}{2}X(\Omega-\Omega_c) + \frac{1}{2}X(\Omega+\Omega_c)$$

Prevent overlapping: if X within $[-\Omega_m, +\Omega_m]$
$\Rightarrow \Omega_c >= 2\Omega_m$

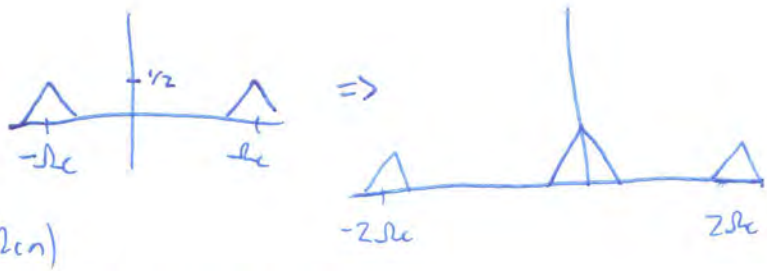
$$r_s[n] = x[n]\sin(\Omega_c n)$$

$$R_s(\Omega) = \frac{j}{2}X(\Omega+\Omega_c) - \frac{j}{2}X(\Omega-\Omega_c)$$

↑ imaginary and real components will swap

Demodulation

- Multiply by a cosine again



$$x[n]\cos(\Omega_c n)\cos(\Omega_c n)$$

$$d[n] = r[n]\cos(\Omega_c n) = x[n]\cos^2(\Omega_c n)$$

$$D(\Omega) = \frac{1}{4}X(\Omega+2\Omega_c) + \frac{1}{2}X(\Omega) + \frac{1}{4}X(\Omega-2\Omega_c)$$

Now take demodulated signal and filter it to get what we want.

Low pass Filter ↓

$$LPF(\Omega) = \begin{cases} 2 & \text{if } \Omega \text{ in } [-\Omega_c, +\Omega_c] \\ 0 & \text{else} \end{cases}$$

$$Y(\Omega) = D(\Omega) * LPF(\Omega) = X(\Omega)$$

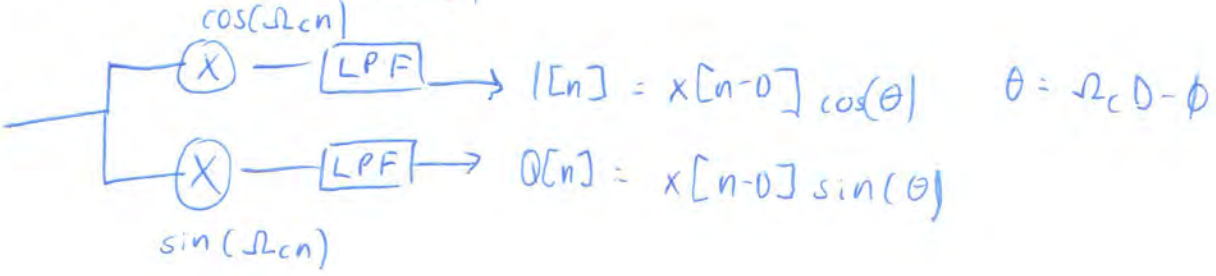
Phase errors

$$d[n] = x[n]\cos(\Omega_c n)\cos(\Omega_c n - \phi)$$

$$\Rightarrow y[n] = x[n]\cos(\phi)$$

↳ could be 0

Solution: Quadrature Demodulation

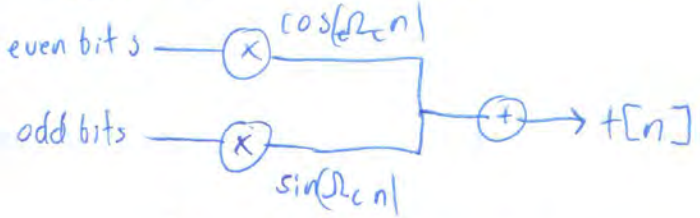


$$w[n] = I[n] + jQ[n]$$

$$|w[n]| = |x[n-D]|$$

QPSK (quadrature phase-shift keying)

- send twice much data



demodulate by cosine to get even bits.
" sin " " odd "

Channel Sharing

- modulate ~~dual~~ signals at different carrier frequency Ω_1 Ω_2

  Pick s.t. $|\Omega_2 - \Omega_1| \geq 2\Omega_m$

  - gap bw frequencies must be at least twice width of spectrum of any signal.

    LPF cutoff should be 1/2 min separation b/w channels

2/2

11/3/15

6.02 Recitation

Fernando Trujano

Any signal can be represented as a signal of cosines and sines

$$x(t) = \cos(2\pi F t + \phi)$$

$$x[n] = x(nT) = \cos 2\pi f n T + \phi) = \cos(\Omega \pi + \phi)$$

$$\Omega = 2\pi F_T = 2\pi F/f_s \Rightarrow f = \frac{f_s \Omega}{2\pi}$$

$x[n]$ from $v(t)$ sampled at $f_s = 1000\ kH_z$

$x_1[n]$ from $x(t)$ sampled at $f_s = 500\ kH_z$

$$X_1(\Omega) = \sum_n X_1[n] e^{-j\Omega n}$$

$$= \sum_n X[2n] e^{-j\Omega n}$$

$$m = 2n \Rightarrow n = m/2$$

$$= \sum_m x[m] e^{-j\Omega m/2}$$

$$= \sum_m x[m] e^{-j \frac{\Omega}{2} m} \Rightarrow X_1(\Omega) = X(\Omega/2)$$

Sample Question

J. Two Tone Code

A) $f_s = 48\ kHz$

max frequency in $x(t) \Rightarrow 24\ kHz$

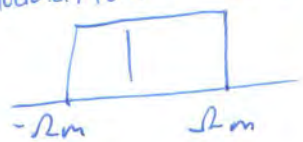
B) $f_1 = 1000Hz \Rightarrow \Omega_1$

$f_2 = 1600Hz \Rightarrow \Omega_2$

$P_1 = \frac{2\pi}{\Omega} = \frac{2\pi}{\Omega_1} = \frac{2\pi F_s}{2\pi F_1} = \frac{f_s}{f_1} = 48$ samples

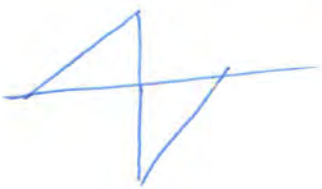
$P_2 = \frac{f_s}{f_2} = 3$ samples

Modulation of cosine



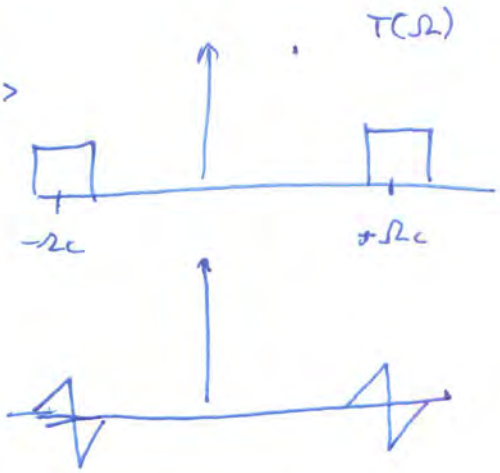
$Re(X(\Omega))$

$t[n] = x[n]\cos(\Omega_c n)$



$Im(X(\Omega))$

$\Rightarrow$



$x[n] \xrightarrow{\times} t[n] = x[n]\sin(\Omega_c n)$ $\sin(\Omega_c n)$ Euler bs

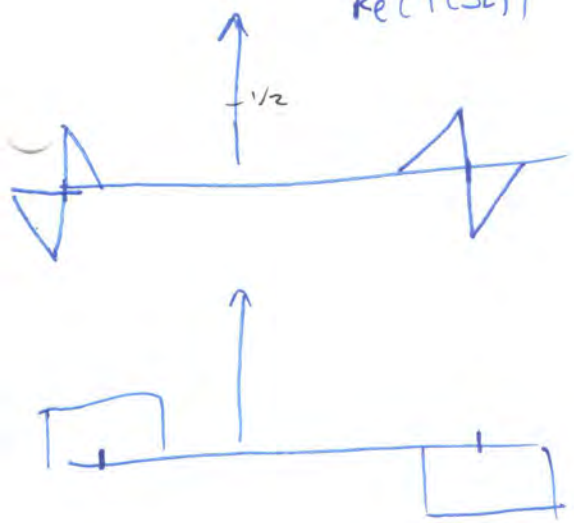
$T(\Omega) = \sum_n \left(x[n]\ \frac{1}{2j}\left(e^{j\Omega_c n} - e^{-j\Omega_c n}\right)\right) e^{-j\Omega n}$

$$= \frac{j}{2} \sum x[n] \left[e^{-j(\Omega+\Omega_c)n} - e^{-j(\Omega-\Omega_c)n} \right]$$

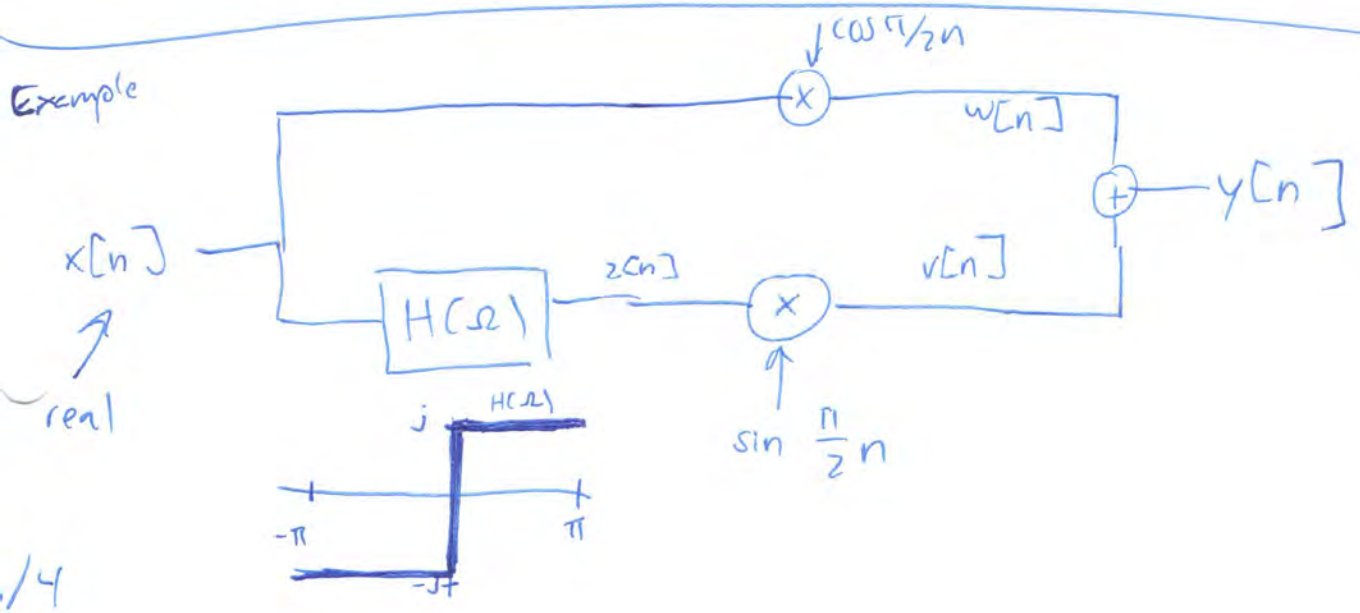
$$= \frac{j}{2} \left[X(\Omega+\Omega_c) - X(\Omega-\Omega_c) \right]$$

multiplied times j → so Real becomes Immaginary !

$$= \frac{j}{2} \left[Re(\Omega+\Omega_c) + j\,Im(\Omega+\Omega_c) - Re(\Omega-\Omega_c) - Im(x(\Omega-\Omega_c)) \right]$$

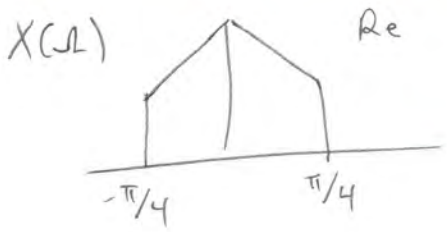


Example



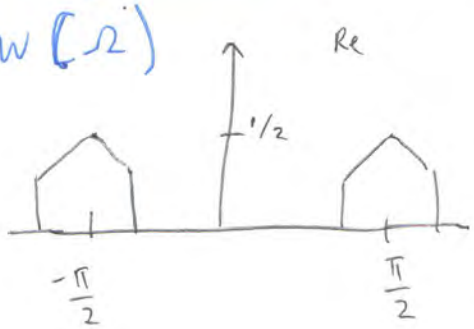
/4

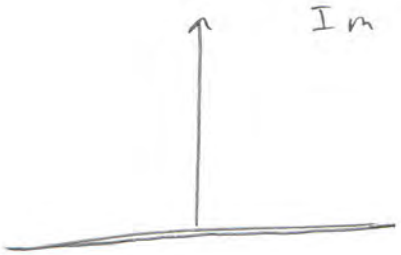
What does $X(\Omega)$ look like?



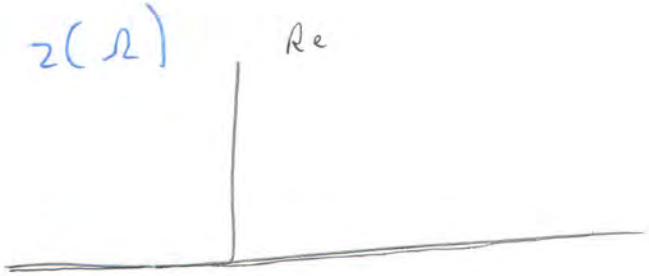
What is spectrum of. modulated by $\cos \pi/2 n$

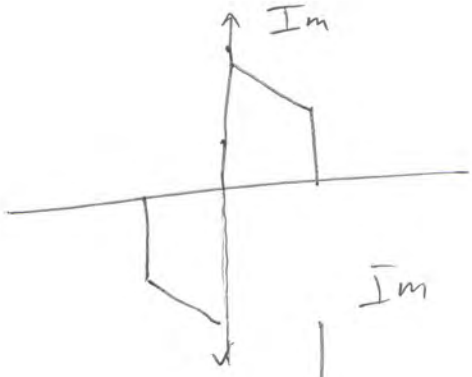
$w(\Omega)$



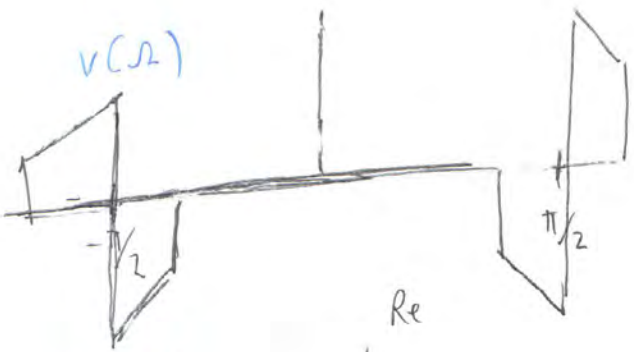


$z(\Omega)$





$v(\Omega)$



$y(\Omega)$

4/4

11/4/15 6.02 Lecture Fernando Tirjano

Analyze Protocols that let nodes share a single channel

1) Collisions happen if two (or more) nodes transmit at once.
2) Nodes know when a collision occurs
3) Nodes <u>might</u> be able to hear each other (sometimes)

Goal: Design a media access protocol Avoid collisions, good performance

1) utilization - use channel efficiency

$$U = \frac{\text{total throughput (data/time) over all nodes}}{\text{max data rate of channel}}$$

↖ should be between $0 \to 1$

Ex:

Channel capacity: 10 Mbit/sec

	Throughput
N_1	1 Mb/sec
N_2	2
N_3	2
N_4	3

$$U = \frac{(1+2+2+3)\text{Mb/sec}}{10\text{ Mb/sec}} = .8$$

2) Fairness: Divided evenly

X_i = throughput that node i achieves

N = # nodes in network

$$F = \frac{\sum_i^N x_i}{N \cdot \sum_i^N x_i^2}$$

$$\frac{1}{N} \leq F \leq 1$$

↖ Perfect fairness

1) Time is divided into slots of equal weight
2) Nodes can start transmitting at beginning of a slot
3) All packets are same size - an int # of slots

TDMA

1) Each node gets a unique index $[0, N-1]$
2) Node i can transmit in timeslot t iff t and $N = i$

- Fair
- Bad if work load is skewed → waste time

Slotted ALOHA

Assume: each packet is 1 slot long

If a node is backlogged, it sends a packet in the next timeslot w/ prob p.

If $p \uparrow\uparrow$ → too many collisions
$p \downarrow\downarrow$ → no good utilization

$$p \cdot (1-p)^{N-1} \cdot \binom{N}{1} = N$$

Stabalized ALOHA.

1. On Success

$$p = \min(2p_i, 1)$$

2. On collission - decrease

$$p = \max\left(p/2_i, p_{in}\right)$$

9/17/15

6.02 Recitation

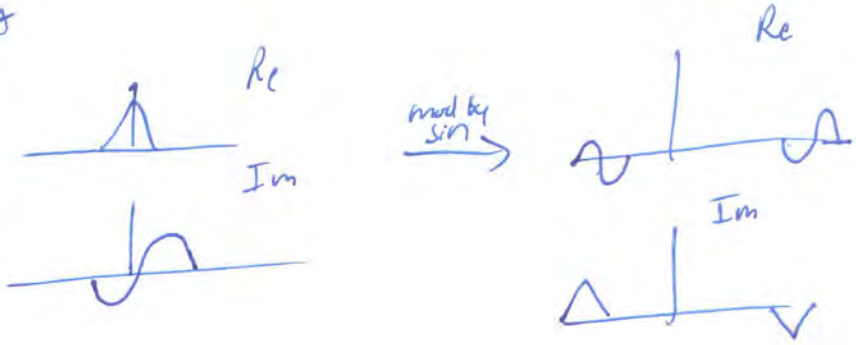
Fernando Trujano

Fourier Transform DTFT

$$X(\Omega) = \sum_m x[m] e^{-j\Omega m} \qquad x[n] = \frac{1}{2\pi} \int_{(2\pi)} X(\Omega) e^{j\Omega n} d\Omega$$

Energy

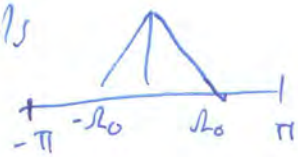
$$BER_{\substack{\text{on-off} \\ 5 \to 0}} = Q\left(\frac{\Delta V}{2\sigma}\right)$$



Why Gaussians?

Central limit Theorem:
averaged time-signals from distribution => Gaussians

-Band limiting signals



↳ Not Aliasing!

Properties that relate $h[n] \to H(\Omega)$

$H(0) = \sum h[n]$ $\quad H(\pi) \sum (-1)^n h[n]$

Modulate with cosine → demodulate w cosine and phase shift

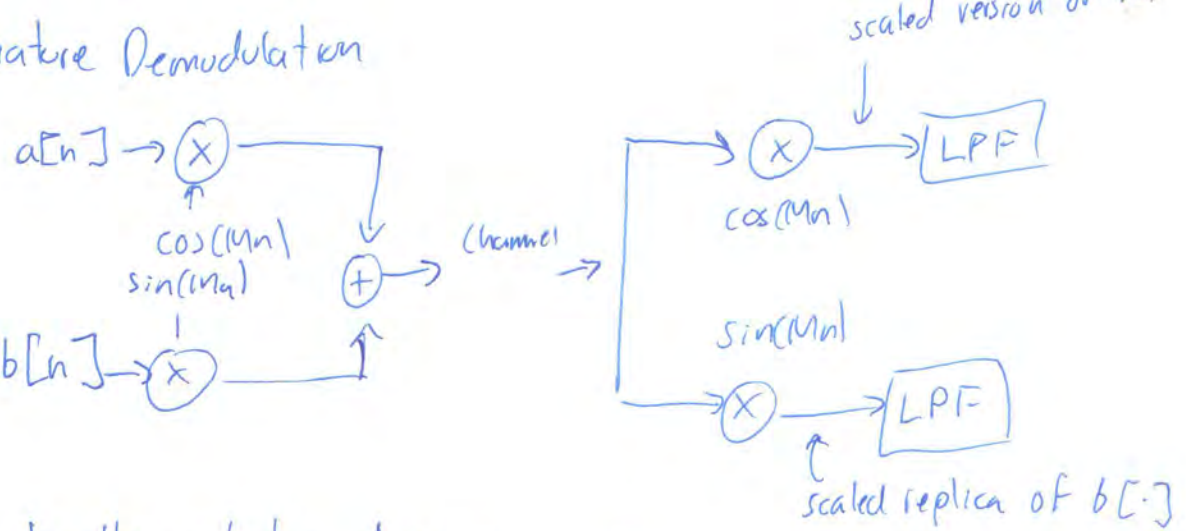
Solution: Demodulate by both cosine and sin.

↳ Quadrature Demodulation.

Modulation

- Demodulate by something with $\pi/2$ phase difference with mod
  => components at origin cancel each other out
  => spectral coefs of $y[n] = 0 \ \forall n$

Quadrature Demodulation



- handle arbitrary delay lag and

frequency: samples/second

Things to believe

$$e^{j\pi} = -1$$

$$y[n] = Y(0) + Y(1)e^{jn} + Y(2)e^{j2n} + \ldots\ldots$$

Frequency of $j = \pi/2$

$-j = 3\pi/2$

$e^{j\pi/2}$

If you see j in $x[n]$ replace $e^{j\pi/2}$

$$\cos\phi = \frac{e^{j\phi} + e^{-j\phi}}{2}$$

$$e^{j\phi} = \cos\phi + j\sin\phi$$

$$\sin\phi = \frac{e^{j\phi} - e^{-j\phi}}{2j}$$

$a + ib$

magnitude

$\angle$ ← angle

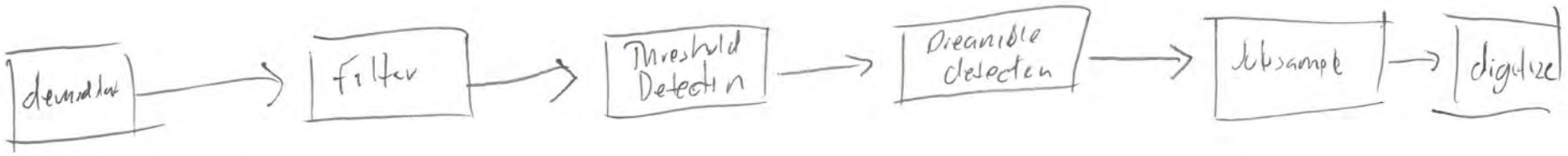
$$\text{amp} = \sqrt{a^2+b^2}\, e^{j\arctan2(b/a)}$$

← complex conjugate

$$H(\Omega) = H^*(-\Omega)$$

Fourier transform

$$\mathcal{F}(\square) = \sum_{n=-\infty}^{\infty} \square\, e^{-j\Omega n}$$



rectify

Absolute values

Take average

What is one what is 0

Find where signal start

Convert 5 samples to single bit

error correction

otherwise avg = 0

6.02 Lecture #17 Notes

Networks

Analyze protocols that allow many nodes to share a single channel.

=> Design a media access protocol - which nodes will use to access the channel that avoids collisions and ensures good performance.

Utilization - using the channel efficiently

$$U_{channel} = \frac{\text{total throughput over all nodes}}{\text{max data rate of channel}}$$

(total throughput over all nodes: amount of data transfered over time, data/time)

$U = 1$ => using 100% of channel

↳ U cannot > 1 or < 0

Fairness - divide capacity equally among nodes requesting to use it.

$$F = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \cdot \sum_{i=1}^{N} x_i^2}$$

N: total # nodes

x_i: throughput of node i

$$1/N \leq F \leq 1$$

$1/N$: one node getting all throughput

1: Perfectly fair

Abstractions:

- each packet is same size
- collisions happen when 2 senders send @ same time
- nodes can detect when collision occured

Time Division Multiple Access (TDMA)

- Give each node a unique index
- Node i can transmit in time slot t iff $t \bmod N = i$
  - Node transmits every N time slots

Analysis:

- No collisions
- Fair if all nodes have data to send
- Utilization
  - Good if every node sends all the time.
    - otherwise => wasted time slots
  - Not good with different loads

ALOHA

Slotted ALOHA - assume each packet takes 1 timeslot to transmit

- if node has stuff to send, send it on the next time slot with probability p.
  - large p => collisions
  - small p => under utilization

$$P(\text{successful\_transmission}) = p \cdot (1-p)^{N-1}$$

($N-1$: # of backlogged nodes; $(1-p)^{N-1}$: all others don't send; p: one sender)

$$U_{\text{slotted\_ALOHA}} = N \, p \cdot (1-p)^{N-1}$$

(N: ways to have exactly one sender)

maximize U: 1) Take derivative = 0 => max => $p = 1/N$

$p = 1/N$ => $U = (1-1/N)^{N-1}$

Problem: N changes so optimal p changes, but nodes can't independently update p.

Solution:

Stabalized ALOHA

- Let nodes dynamically adjust p.

1) IF transmission collides set $p = p/2$ ← multiplicative decrease

2) IF transmission works set $p = \min(2p, 1)$

Problem: If a node has several failed attempts p will be really low and it will be hard to come back.

Solution:
1) If fail: $p = \max(p_{min}, p/2)$
2) If YES: $p = \min(2p, 1)$

Problem: Nodes can capture a channel and transmit in burst, which is unfair to others at that time. This is because p can get really high.

Solution:
1) On Fail: $p = \max(p_{min}, p/2)$
2) On success: $p = \min(2p, p_{max})$

CSMA - Carrier Sense MuHiple Access

If nodes can hear each other, check if someone is transmitting before transmitting. → carrier sense

More in P57

modify stabalized ALOHA

Other Goals

1) Upper bound on wait to make transmission
2) Handle variability in node work. (How often they send)
3) Scalability

6.02 Lecture 18 Notes

Fernando Trujano

Introduction to Networking

Circuit Switching

Protocol

1) Setup-Phase: configure state with mappings in switch ← don't know destination, only where to send next
2) Data-Transfer-Phase: Switches forward data
3) Teardown-Phase: Delete phase in switch

To share same link use TDM. (Time Division Multiplexing)
↳ very similar to TDMA

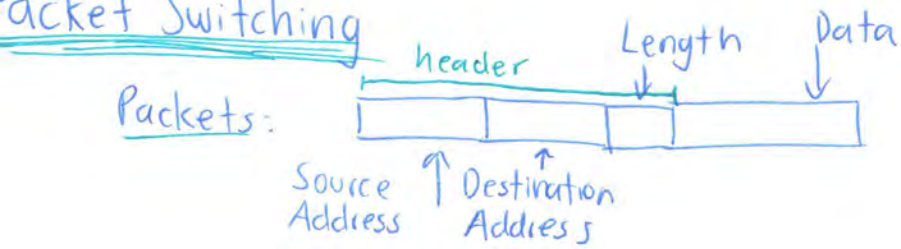
- Divide time into N slots "packet"
  - Slot 0: send frame for conversation 0 ....

Problems w/ TDM: ← like TDMA

1) Bad when data arrives in bursts
2) Bad when communications send more/less than R bits/sec (what each connection requires) or have more than C/R communications. (C: what the link we are sharing can handle)
3) If a link in middle of network goes down, all circuits using that link are broken.

Packet Switching

Packets:



← knows destination

More resiliant to Failure, (fixes #3 in Circuit switching)
↳ packet can take longer route if node missing.

Switching: Store incomming packets in queue (drop if queue full). Determine outgoing link of first packet using header.

Benefits:

1) Links can send any available packet => no wasted link capacity
2) Variable data rates "as needed"
3) No setup / teardown phase
4) Aggregating multiple sources of traffic smooths out bursts (Queues absorb bursts!)

Bad: 1) Packets can be dropped 2) Packets may arrive in different order
3) No guarantee on latency

Queues

Queues add more delays:

Queuesing delay: How long packets waits in queue

Processing delay: time spent by switch to process packet

Little's Law

↳ Average time packet spends on queue

$N = \lambda \cdot d$

(λ: average rate of switch; N: mean # of packets in Q.; d: mean delay per packet)

Circuit Switching

- Guaranteed Rate
- wasted link capacity if bursty
- Path established before sending data
- All data in connection Follows 1 path
- No reordering; constant delay, no dropped packets
- can't route around Failures

Packet Switching

- No guarantees
- More efficient
- Send data immediatly
- Different packages could follow different paths
- Packes may be reordered, delay or dropped
- can route around Failures

6.02 Lecture #19 Notes

Addressing, Forwarding, Routing w/ Failures

How do nodes determine routes to destination?
↳ allow each switch to know how to get to any other node
↳ with minimum cost.

Addressing: How to name a node. Addresses are globally unique.

Costs: Links can have costs/weights: 1) Non-negative 2) Triangle inequality
3) Can change
$cost(A \to B) + cost(B \to C) \geq cost(A \to C)$

Routing Tables: Table to store information about routes. ← and sometimes, cost
If packet is destined for X, where to send the packet next.

Forwarding: When switch receives packet:
1) Check packet header for destination address, dest
2) Look up routing_table[dst] to get out going link
3) Add packet to queue
4) wait .... and send when in front.

Distributed Routing

↑ each node runs routing protocol and determines own routes

General Structure:
1) Nodes learn about neighbors via HELLO structure
   - Nodes send HELLO packets to each neighbor
2) Nodes learn about other nodes via advertisements
   - Sent by other nodes containing information about other nodes
   - use advertisements to determine connectivity and cost ↑ to reachable nodes
3) Determine min-cost routes with information gathered
   ↳ "integrate" step

Difference b/w distance vector and link-state comes in
- Types of advertisements sent and who they are sent to
- Way nodes integrate advertisements

Distance-Vector routing protocol

"Talk to neighbors about everyone"

Advertisement: Each nodes sends: (dst, cost)
to: only its neighbors

Integrate step: Bellman Ford
For each (dst, cost):
If X is already using Y to get to dst: update cost
else if Y can provide better path to dst: update routing and cost

advertisement from Y→X

Link State

"Talk to everyone about neighbors"

Advertisements: Each node sends:
List of neighbors and link cost to neighbors "flooding"
to: neighbors who then forward to neighbors ... => All network

Integrate step: Run Dijkstra on now complete map of network ← relax edges, find all shortest paths → O(NlogN + E)

Note that advertisement only changes if neighbor link cost changes

6.02 Notes

Lecture #20
Routing with Failures

2. What happens when parts of the network fail and how do nodes deal w/ that.

Distance Vector : Each ad sent to neighbors => 2L ads DVR less bdwd
Link State : Flood network. => N2L ads

Protocol Correctness

- routing tables should reflect correct network topology

1) Route validity: ∀ N if dst in routingTable and routingTable[dst] = k
   => There is path from N to dst whose first hop is k.
   ↳ invalid => disconnected network info not yet propagated Route loops

2) Path visibility. Every router that has a path to a dst, learns at least one valid route to dst.
   ↳ route valid + visible

Eventual convergance: routing protocol will eventually converge at t, could fail in between

Link state routing failures

- If no ads are lost => protocol will converge
  lost ads can lead to routing loops but rare b/c flooding
  Link state => Fast convergence

Distance Vector Routing Failures

Recall in DVR nodes don't know entire topology

- Cout to Infinity Problem
  - Attempt #1: Split Horizon
    - Dont send path to dst to neighbor through wich you would go
    - Avoids only two-hop loops
  - Attempt #2: Path vector routing
    - Include min cost path in add. Node N will integrate only if N not in path,
    - converges and does not count to infinity

12

Distance Vector: uses less bandwidth
- use for very small networks
  ↑ loop-Free

Link-State: convergest quickest, but large overhead
- use for small networks

Path-Vector
- use for large networks

11/24/15

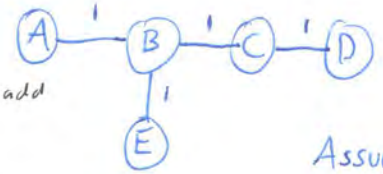
6.02 Recitation

Fernando Trujano

1. Distance Vector Routing: Count-to-infinity
2. Link State Routing: Routing Loops
3. Chapter 18 Problems 5, 8, 10, 11

(1)

Count to Infinity



E: [A, 2; B, 1; C, 2; D, 3; E, 0] ← what E would send to neighbors

Assume long time passed, everyone has it's routing table.

Assume new link costs get updated if < old.

before A sends add

Order of Ads

i) A
ii) B
iii) C
iv) D
v) E

=> A↔B count to inf

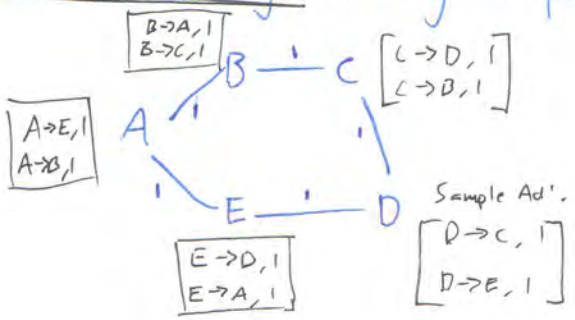
← No count-to-infinity

↖ link BE fails => B↔C count to infinity.

EVENT	ACTION
① BE Fails	@B E: B→E, ∞ @C E: C→B, 2 (total cost B→C) @A C: A→B,2
② C sends ad [A,2; B,1; C,0; D,1; E,2]	@B E: B→C, 3 ← D thinks C can get to E in 2 steps, updates its path to go through C.
D sends add ..	
E sends add	
③ A sends ad [A,0; B,1; C,2; D,3; E,2]	@B E: B→C, 3
④ B sends ad [A,1; B,0; C,1; D,2; E,3]	@C E: C→B, 4 @A E: A→B,4

/2

(2) Link State Routing - Routing Loops



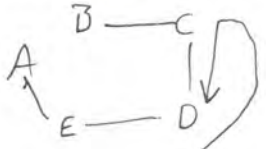
Ad's flooded through network

C receives

A→C lost

B→C lost

=> C thinks network looks

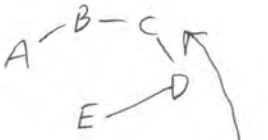


D receives

A→D lost

E→D lost

=> D thinks network looks like this

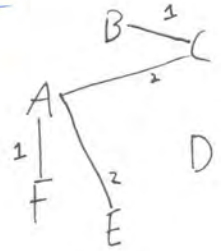


now if D wants to send to A.
sends it to C first
C sends it back to D

Chapter 18 Problems

Problem 5

a)

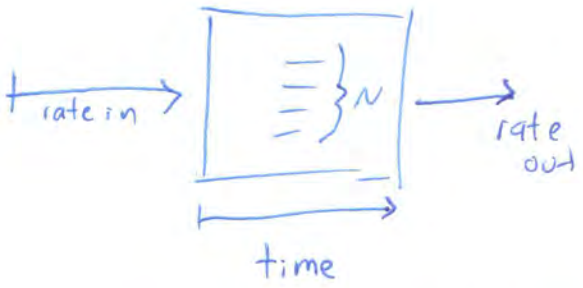


b) D-> F, 3

6.02 Recitation

Little's Law

- Abstract any queueing system



N - [packets] - average # in queue

λ - [packets/time] - flow rate

D - [time] - average delay time of packet in queue.

$$N = \lambda D$$

$$\sum_{t=0}^{T} n(t)/T = \lambda \sum_{t=0}^{T} n(t)/P$$

TC

Fernando Trujano

6.02 Recitation

12/1/15

Reliable Data Transport Protocols

Stop and Wait

- unique ID for each packet, Sender sends K
- Receiver sends ACK
- Sender sends K+1

* ~~Could have dupl~~

- Wait some time to receive ACK, otherwise send again.
  ↳ could lead to duplicates

Reliable =
- all data
- in order
- one copy

Examples from chapter 19

See notes

FERNANDO TRUJANO 6.02 QUIZ I

Information and Entropy

Probability

- P of mutually exclusive events adds: $P(A \text{ or } B) = P_A + P_B$
- If not mutually exclusive: $P(A \text{ or } B) = P_A + P_B - P(A \text{ and } B)$
- P of independent events multiply

Conditional Probability

$P(A|B) = P(A \text{ and } B) / P(B)$

If $P(A|B) = P(A)$ => A, B independent => $P(A \text{ and } B) = P_A P_B$

$P(A \text{ and } B) = P(A|B)P(B) = P(B|A)P(A)$

Bayes Rule: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Information

Expected length = $\sum_\alpha P_\alpha R_\alpha$

I_α = information associated with revealing output = α

$I_\alpha = \log_2(1/P_\alpha)$ bits $= -\log P_\alpha$

H = entropy = $\sum_\alpha P_\alpha I_\alpha = \sum_\alpha P_\alpha \log_2(\frac{1}{P_\alpha})$

↳ expected info over all outcomes

Example: A, B independent

$I_A + I_B = \log(1/P_A) + \log(1/P_B) = \log(1/P_A P_B) = \log(1/P(A \text{ and } B))$

Example: p = P of bit flip. Codeword N bits

$P(>1 \text{ error}) = 1 - p(\text{no error}) - p(1 \text{ error})$
$= 1 - (1-p)^N - Np(1-p)^{N-1}$

- Go from N equally-probable outcomes to M outcomes: $I = \log_2(N/M)$
- Entropy = lower bound on expected length of messages
- Go from H_1 to H_2. Info revealed $= H_1 - H_2$
- ↓ probable event => ↑ info revealed

Source Coding

Source codes - remove redundancy and compress data.

Entropy H is a lower bound on amount of info that must be sent.

Prefix Free => all symbols at the end of tree

Huffman Encoding

- Know probabilities in advance
- More probable = shorter codeword

Algorithm:

- sort nodes on p_i
- merge two least probable
- combine and repeat till done

LZW compression

- No need to know probabilities in advance

Encoding

```
table = initialize_table()   // 0 to 255
                             // ascii
string = get input symbol
while input symbols:
    symbol = get input symbol
    if string + symbol in table:
        string = string + symbol
    else:
        output code for string
        table.add(string + symbol)
        string = symbol
output code for string
```

Decoding:

```
table = initialize_table()
code = read next code
string = table[code]
output string
while codes:
    code = read next code
    if not table[code]:
        entry = string + string[0]
    else:
        entry = table[code]
    output entry
    table.add(string + entry[0])
    string = entry
```

- If the dictionary fills up, empty it before adding new element
- Just because string s was added to the table does not mean it was sent, but all prefixes of s have

Errors, Binary Symmetric Channels
Hamming Distance

Code rate = $\frac{\text{\# useful message bit}}{\text{\# total bits sent}}$

Hamming Distance = distance b/w codewords of equal length

$H(A,B) = A + B \bmod 2$

Detect t errors if: $MHD \geq t + 1$ => Errors detected = MHD − 1

Correct t errors if: $MHD \geq 2t + 1$ => Errors corrected = $\lfloor \frac{MHD-1}{2} \rfloor$

Linear Block Codes (n, k, d)

d = MHD

Block codes: message -> k bit blocks -> n bit codeword. Rate = k/n

Systematic codes: [message bits (k) | parity bits (n−k)]

Linear Block Codes: sum of 2 codewords = another CW

Obtain CW by doing linear trans of message bits with G

codeword, i^{th} elm row of data, k elm row

↳ $C = dG$ ← k×n generator matrix

$G = [I | A]$ ← parity spec matrix

i^{th} entry in A weather d_i was used in constructing p_j

Rectangular Codes

- calculate parity bit of each row r and column c

d_1	d_2	p_1
d_3	d_4	p_2
p_3	p_4	

Rate = $\frac{rc}{rc + r + c}$

$G = I$ with rows 1010, 1001, 0110, 0101 (columns $p_1 p_2 p_3 p_4$) ← d_1 protected by p_1, p_3

Not efficient so =>

Hamming Codes

- Correct single bit errors with min # of parity bits. $n = 2^{n-k} - 1$

$P_1 = d_1 + d_2 + d_3$ -> protects all bits w/ least sig dig set

$P_2 = d_1 + d_3 + d_4$ " second "

$P_3 = d_2 + d_3 + d_4$ " third "

- Code rate of block code with 2^k codewords is k/n ← # bits in each codeword
- If a linear block code can correct all single bit errors, then its parameters n, k must satisfy $2^{n-k} \geq n + 1$

For t errors: $2^{n-k} \geq 1 + \binom{n}{1} + \binom{n}{2} + \binom{n}{3} + \dots \binom{n}{t}$

- There always exists a (n, n−1, 2) code => Adding parity bit to each string
- 0 codeword always in a LBC
- MHD = weight of min(weight) non zero codeword
- Code word d is odd + parity bit => MHD ++ => can correct 1 more error
- Code word d is even + parity bit (not => MHD ++ but can detect 1 more error (correct)
- 2^n possible codewords, 2^k valid ones

Syndrome Decoding

- Efficient way to decode any linear block code.

$H r^T = S$ (syndrome bits), $H = [A^T | I]$ ← r: received code word

Error vectors -> have 1 where there is an error. For Single Error Correction $e_i = I_i$ so

He_i = i^{th} column of H

Algorithm

1) For each error pattern e_i compute syndromes $He_i^T = S_i$ and add to syndrome table
2) For each received word compute $Hr^T = S$
3) Find l such that $S_l = S$ and correct error $C = R + e_l$ ← flip bits wher 1 on e_l
4) Decode the codeword by taking first k bits

- The MHD of code = # linearly dependent cols of H ie can be combined to give zero vector.
- A has identical rows => H has identical cols => MHD = 2 => error correction not possible

---

Ex: $P_1 = D_1 + D_2 + D_3$
$P_2 = D_2 + D_3$
$P_3 = D_1 + D_3$

- To find MHD, list all code words (all combinations of 3 data bits and corresponding parity bits)
- Decode 101101
  1) Compute Syndromes $E_1 = D_1 + D_2 + D_3 + P_1 = 1$, $E_2 = D_2 + D_3 + P_2 = 1$, $E_3 = D_1 + D_3 + P_3 = 1$ $= [111]$
  2) D_3 appears in all three -> flip it
  3) Return first k bits => 100

Convolutional Codes

- Work on continuous stream of bits
- Transmit only parity bits

Encoding:

$G_0 = 111$ => $P_0 = x[n] + x[n-1] + x[n-2]$
$G_1 = 101$ => $P_1 = x[n] + x[n-2]$

Message: 11001101 (pad w/ zeroes) => 0011001101 0

$x[0]\ x[1]$, $x[-1]\ x[0]$

$P_0[0] = 1$, $P_1[0] = 1$
$P_0[1] = 0$, $P_1[1] = 1$
$P_0[2] = 0$, $P_1[2] = 1$
... ...

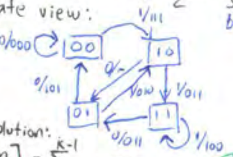
Transmit: 110101...

K = 3: code's constraint length

Looking at message with a "sliding window" of length K
↳ each message bit protected by K parity bits

- Encoder can be in 2^{K-1} states

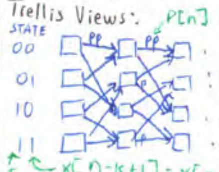
State view:



bit read / parity bits transmitted

Convolution: $P_i[n] = \sum_{j=0}^{K-1} g_i[j] \times [n-j]$ ← input bitstream as function of j flipped in time

Trellis Views:



$x[n-1]$, $x[n-K+1] = x[n-2]$

- label arrows w/ parity bit sent
- Upper arrow => 0 received, lower arrow => 1 received
- Any path through trellis is a code word

Encoding: For each message bit, make transition and send corresponding parity bits. End message w/ K−1 0's to return to starting state

Free Distance: Smallest weight nonzero codeword = dfree

Can correct $\lfloor \frac{dfree - 1}{2} \rfloor$ errors if they are "far enough apart"

Viterbi Decoding

- Find most likely path that transmitter took

Branch Metric − HD b/w received parity bits and parity bits of transitions
↳ How likely we did NOT make that transition

Path Metric - smallest sum of BM minimized over all message.

Algorithm: Trace back by hand

For each state s at time i+1

a, b = predecessor-states(s)

$BM_a = BM(a)$, $BM_b = BM(b)$

$PM[s, i] = \min(PM[a, i] + BM_a, PM[b, i] + BM_b)$

Keep track of predecessor to get to s

Free Distance - Find by finding smallest weight path that leaves 0 and returns to 0
↳ how long it takes the decoder to "settle"

Puncturing - Agree to only send a subset of parity bits. Increases Rate

Noise - PDF

Probability Density Function $F_W(w)$

$P(w_1 < W \le w_2) = \int_{w_1}^{w_2} F_W(\alpha)\,d\alpha$

Mean $= N_w = \int_{-\infty}^{\infty} \alpha f_W(\alpha)\,d\alpha$

variance $= \sigma^2 = \int_{-\infty}^{\infty} (\alpha - N_w) F_W(\alpha)\,d\alpha$

$Q(w)$: Probability of Gaussian RV taking values greater than w.

$Q\left(\frac{w-\nu}{\sigma}\right)$: P. of taking vals $> w$. For Gaussian distribution

Bit Error Rate (BER)

↳ P. that any given bit is in error

$BER = Q\left(\frac{V_1 - V_0}{2\sigma}\right)$ $BER_{bipolar} = Q\left(\frac{V_P}{\sigma}\right)$

so: $BER_{on\text{-}off} = Q\left(\frac{V_P}{2\sigma}\right)$

Signal to Noise Ratio SNR

signal power / noise variance

- Average k samples in bit slot
- $N=0$ still but σ^2/k instead of σ^2
- $\Rightarrow$ SNR ↑ by factor K.

On-off: $0 \to 0$ ν, $1 \to 1$ V — Bi-polar: $-V \to 0$, $V \to 1$

Single sample $BER = Q(V/\sigma)$

M samples $BER = Q(\sqrt{M}V/\sigma)$ — variance σ^2/M, var $= \sigma^2$

bit energy in bit slot = SUM OF squares OF M values sent. $Energy_{on\text{-}off} = MV^2/2$

$Energy_{bipolar} = MV^2$

LTI Models

↳ Linear - Time Invariant

$x[n] \to [S] \to y[n]$ (input, output)

Causal Model: $y[n]$ only depends on past and present values of $x[\,]$

Unit Step: $u[n]$ — Unit sample $\delta[n]$

$\delta[n] = u[n] - u[n-1]$

$u[n] \to [S] \to s[n]$ (unit step response) $\quad \delta[n] \to [S] \to h[n]$ (unit sample response)

Can characterize LTI system with ↑

$x[n] = \sum_{k=-\infty}^{\infty} x[k]\,\delta[n-k]$ (CONVOLUTION)

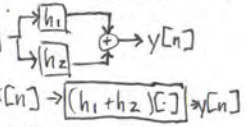
$\Rightarrow y[n] = \sum_{k=-\infty}^{\infty} x[k]\,h[n-k]$ (h reversed in time, shifted in to the right.)

$y[n] = x * h = h * x \Rightarrow y[n] = \sum_{m=-\infty}^{\infty} h[m]\,x[n-m]$

Series: $x[n] \to [h_1] \to [h_2] \to y[n]$

$= x[n] \to [h_1 * h_2] \to y[n]$

Parallel:



$= x[n] \to [(h_1 + h_2)[\,]] \to y[n]$

$h[n] \to s[n]$: $s[n] = \sum_{k=-\infty}^{n} h[k]$

Period $= 2\pi/\Omega_0$

Frequency Response

Periodic: $x[n+P] = x[n]$

smallest P = period

Periodic input → Periodic outputs

Freq Ω → Freq Ω

Discrete Time Sinusoid

$x[n] = \cos(\Omega_0 n + \theta_0)$

$\Omega_0 = 0 \Rightarrow$ lowest variation

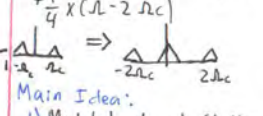
$\Omega_0 = \pm\pi \Rightarrow (-1)^n \Rightarrow$ highest "

- All action of interest happens at the $[-\pi, \pi]$ range.

Complex Exponentials

$e^{j\phi} = \cos\phi + j\sin\phi$ (Euler)

ϕ = angle in unit circle



$x[n]$ is a sum of complex exp.

$\cos\phi = \frac{1}{2}\left(e^{j\phi} + e^{-j\phi}\right)$

$\sin\phi = \frac{j}{2}\left(e^{-j\phi} - e^{j\phi}\right)$

$\cos(A)\cos(B) = \frac{1}{2}(\cos(A+B) + \cos(A-B))$

$\cos(A \pm B) = \cos(A)\cos(B) \mp \sin(A)\sin(B)$

★ $H(\Omega) = \sum_{n=-\infty}^{\infty} h[n]\,e^{-j\Omega n}$ DTFT (spectrum, spectral content)

$h[n] = \frac{1}{2\pi}\int_{\langle 2\pi \rangle} H(\Omega)\,e^{j\Omega n}\,d\Omega$

$H(0) = \sum_{n=-\infty}^{\infty} h[n]$ $\quad H(\pm\pi) = \sum_{n=-\infty}^{\infty} (-1)^n h[n]$ max-frequency

$h[n] = (h_1 * h_2)[n] \Rightarrow H(\Omega) = H_1(\Omega)\,H_2(\Omega)$ (convolution in time $\Rightarrow$ multiplication in frequency)

$h[n] = (h_1 + h_2)[n] \Rightarrow H(\Omega) = H_1(\Omega) + H_2(\Omega)$

IF $h_0[n] = h[n-D]$ then $H_0(\Omega) = e^{-j\Omega D}$ and $|H_0(\Omega)| = |H(\Omega)|$, $\angle H_0(\Omega) = -\Omega D + \angle H(\Omega)$

↳ shift in time $\Rightarrow$ Linear phase frequency (addition in $H(\Omega)$)

$Y(\Omega) = H(\Omega)\,X(\Omega)$

$y[n] = Y(0) + Y(1)e^{jn} + Y(2)e^{j2n} + \dots$

Ex: $h[n] = 2\delta[n-1] + \delta[n-2] + -2\delta[n-3]$

Find $H(\Omega)$ in $e^{-j\Omega D}(1 + jz(\Omega))$ form

1) $H(\Omega) = \sum_{-\infty}^{\infty} h[n]e^{-j\Omega n}$

$= 2e^{-j\Omega} + e^{-j\Omega 2} - 2e^{-3j\Omega}$

$= e^{-j\Omega 2}\left(2e^{j\Omega} + 1 - 2e^{-j\Omega}\right)$ (factor)

$= e^{-j\Omega 2}\left(\frac{j}{j}\left[2e^{j\Omega} - 2e^{-j\Omega}\right] + 1\right)$ (multiply and divide by $-j/2$)

$= e^{-j\Omega 2}(1 + j4\sin(\Omega))$

Assume at Ω_0 $1 + jz(\Omega) = Ae^{j\theta_0}$

Find $y[n]$ given $x[n] = \cos(\Omega_0 n)$

$H(\Omega_0) = e^{-j\Omega 2} A_0 e^{j\theta_0} = A_0 e^{j(-2\Omega + \theta_0)}$

So $|H(\Omega_0)| = A_0$ (scale ↑) $\quad \angle H(\Omega_0) = \theta_0 - 2\Omega$ (shift ↑)

$\Rightarrow y[n] = A_0\cos(\Omega_0 n + \theta_0 - 2\Omega)$

DTFT of $\delta[n] = 1$ $\quad \sum_{-\infty}^{\infty} jK$ $\quad \frac{1}{\quad}$ for complex λ

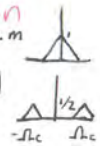
Modulation Demodulation

Modulation

$X(\Omega) = \sum x[n]e^{-j\Omega n}$

$r[n] = x[n]\cos(\Omega_c n)$

$R(\Omega) = \frac{1}{2}X(\Omega - \Omega_c) + \frac{1}{2}X(\Omega + \Omega_c)$



Modulating by sin

$r[n] = x[n]\sin(\Omega_c n)$

$R_s(\Omega) = j/2\,X(\Omega + \Omega_c) - j/2\,(\Omega - \Omega_c)$

Demodulation - Multiply by cosine again

$x[n]\cos(\Omega_c n)\cos(\Omega_c n)$

$d[n] = r[n]\cos(\Omega_c n) = x[n]\cos^2(\Omega_c n)$

$D(\Omega) = \frac{1}{4}X(\Omega + 2\Omega_c) + \frac{1}{2}X(\Omega) + \frac{1}{4}X(\Omega - 2\Omega_c)$

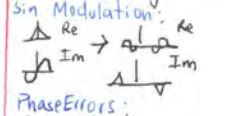
$\Rightarrow$

Main Idea:

1) Modulate to shift the spectrum of input signal to match freq response of channel.

2) Recover signal by demodulation and Filtering.

Sin Modulation:

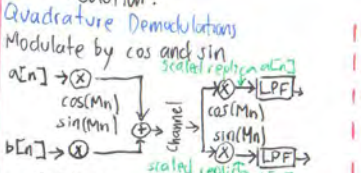


Phase Errors:

Demodulating with a cosine w/ a phase shift. Solution:

Quadrature Demodulations

Modulate by cos and sin



- Can handle arbitrary delay lag

Quadrature Phase-Shift Keying (QPSK)

send twice as much data by modulating even bits by sin and odd by cosine.

Frequency division Multiplexing (FDM)

- Lets share channel amongst multiple senders.
- Modulate by different frequencies far enough apart

$-3e^{j\pi/2} = 3e^{j3\pi/2}$

IF magnitude of $H(\Omega)$ can be negative, you need to reverse the angle by adding $\pm\pi$.

IF $h[n]$ is even function of time $\Rightarrow H(\Omega)$ purely real. $h[-n] = h[n]$

IF $h[n]$ is odd fn of time $\Rightarrow H(\Omega)$ purely immaginary. $h[-n] = -h[n]$

Deconvolution Filter

$(h_2 * h_1)[n] = \delta[n] \Rightarrow H_2(\Omega) = 1/H_1(\Omega)$

Complex Numbers $c = a + jb$

$|c| = \sqrt{a^2 + b^2}$ $\quad \angle c = \arctan(b/a)$

$a = |c|\cos(\angle c)$ $\quad b = |c|\sin(\angle c)$

Polar Form: $c = |c|e^{j\angle c}$ (angle, magnitude)

$e^{jA} * e^{jB} = e^{j(A+B)}$

$e^{j\Omega n} \to [H(\Omega)] \to H(\Omega)e^{j\Omega n}$

Magnitude of sinusoidal input scaled by

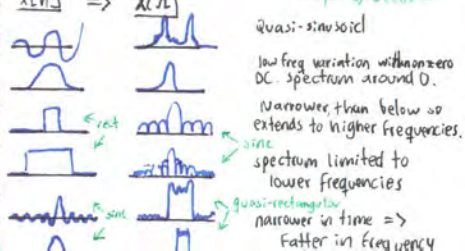
DTFT Matching

1) $X(0) = \sum x[n] \Rightarrow$ IF $X(0) = 0$ $x[n]$ has all signals cancelling out.

2) $X(\pi) = \sum(-1)^n x[n] \Rightarrow$ $\quad$ Period $= 2\pi/\Omega$

$\Omega = \frac{2\pi F}{F_s}$ (F: continuous time, F_s: discrete time) $\Rightarrow \Omega$ is in range $[-\pi, \pi]$ (samples/cycle)

so when $\Omega = \pi \Rightarrow f = f_s/2$ (cycles/second)



Quasi-sinusoid

low freq variation with nonzero DC spectrum around 0.

Narrower than below so extends to higher frequencies.

spectrum limited to lower frequencies

narrower in time $\Rightarrow$ Fatter in Frequency

G = Given
W = work
E = explanation

$x[n] = 2\delta[n-1]$

$y[n] = \delta[n-2] + 2\delta[n-4] + \delta[n-6]$

a) Find $h[n]$: Note that we can get $\delta[n]$ by scaling $x[n]$ by $1/2$ and shifting 1 timestep. We also know that $\delta[n] \to [S] \to h[n]$ so,

$h[n] = .5\delta[n-1] + \delta[n-3] + .5\delta[n-5]$

b) Find $H(\Omega)$ by taking DTFT of $h[n]$:

We know that a shift in time $\Rightarrow$ Linear phase addition in frequency.

So if $h_0[n] = h[n-D] \Rightarrow H_0(\Omega) = e^{-jD\Omega} H(\Omega)$

$H(\Omega) = .5e^{-j\Omega} + e^{-j3\Omega} + .5e^{-j5\Omega}$

c) Find $H(\Omega)$ by taking DTFT $y[n]$ and $x[n]$

$X(\Omega) = 2e^{-j\Omega}$ $\quad Y(\Omega) = e^{-j2\Omega} + 2e^{-j4\Omega} + e^{-j6\Omega}$

$Y(\Omega) = X(\Omega)H(\Omega) \Rightarrow H(\Omega) = Y(\Omega)/X(\Omega)$

$H(\Omega) = .5e^{-j\Omega} + e^{-j3\Omega} + .5e^{-j5\Omega}$

d) Rewrite $H(\Omega)$ in form: $H(\Omega) = e^{jd(\Omega)} A(\Omega)$

1) Factor out an $e^{-j\cdot}$ so that two remaining $e^{\cdot}$ have the same $\angle \Leftarrow 2\Omega$ in this case

$H(\Omega) = e^{-j3\Omega}\left(\frac{1}{2}e^{2j\Omega} + 1 + \frac{1}{2}e^{-j2\Omega}\right)$

$= e^{-j3\Omega}\left(1 + \frac{1}{2}\left(e^{2j\Omega} + e^{-j2\Omega}\right)\right)$

2) "wiggle" equation to substitute a sin or cosine. $\cos\Phi = \frac{1}{2}e^{j\Phi} + e^{-j\Phi}$

$H(\Omega) = e^{-j3\Omega}(1 + \cos(2\Omega))$

So, $\alpha(\Omega) = -3\Omega$ $\quad A(\Omega) = 1 + \cos(2\Omega)$

Note that $A(\Omega) = |H(\Omega)|$ since $A(\Omega) \geq 0 \; \forall \Omega$

e) When is $H(\Omega) = 0$?: when $\cos(2\Omega) = -1$ which happens at $\Omega = \pm \pi/2$

$H(0) = 2$ $\quad x[n] = 1 + (-1)^{n+5}$

$H(\pi) = -3$ $\quad$ Find $y[n]$

We know: $Y(\Omega) = H(\Omega)X(\Omega)$

Step 1: Find $Y(0)$

$Y(0) = H(0)X(0)$ ($H(0) = 2$)

Step 1.1: Find $X(0)$

- look at [component] in $x[n]$ with $\Omega = 0$

$x[n] = [1] + (-1)^{n+5}$

- Rewrite in complex exp.

$1 = e^{j0}$ - Get amplitude

$|e^{j0}|_{Y(0)=2} = 1 = X(0)$ Inverse DTFT

Step 2: $Y(0) \Rightarrow y[n]$

Recall that:

$y[n] = Y(0) + Y(1)e^{jn} + Y(2)e^{2jn} \ldots$

$y[n] = Y(0)e^{j0} + \ldots$

$= 2 + \ldots$

Step 3: Get $Y(\pi)$

$Y(\pi) = \underbrace{H(\pi)}_{-3} X(\pi)$

Step 3.1: Find $X(\pi)$

- look at [component] in $x[n]$ with $\Omega = \pi$ $\quad x[n] = 1 + [(-1)^{n+5}]$

- rewrite in complex exp

$e^{j\pi} = -1 \Rightarrow \left(e^{j\pi}\right)^{n+5} = e^{jn\pi} e^{j5\pi}$

$e^{j5\pi} = -1$ & $\frac{1}{e^{j\pi}} = -1$ $\quad X(\Omega) = 1$ (Amplitude)

$Y(\pi) = H(\pi)X(\pi) = (-3)(-1) = 3$

Step 4: Inverse DTFT

$y[n] = Y(0)e^{jn0} + Y(\pi)e^{jn\pi}$

$\Rightarrow y[n] = 2 + 3e^{jn\pi}$

Networks

Utilization = $\frac{\text{total throughput}}{\text{max data rate}}$

$U=1 \Rightarrow$ 100% utilization

Fairness = $\left(\sum_{i=1}^{N} x_i\right)^2 / N\sum_{i=1}^{N} x_i^2$

(x_i: throughput of node i; N: total # nodes)

Protocols

TDMA - Each node has index i; transmits on timeslot $t \bmod N = i$
- No collisions
- Fair if all nodes have data to send.
- Utilization bad w/ skewed workload

Slotted Aloha If node has stuff to send, send on next t with prob P.

$P(\text{successful transmission}) = p \cdot (1-p)^{N-1}$ ($N-1$: # backlogged nodes; one node sends; others don't)

$U = Np \cdot (1-p)^{N-1}$ $p = 1/N \Rightarrow U = \left(1-\frac{1}{N}\right)^{N-1}$

Stabalized ALOHA: Let nodes dynamically adjust p.
1) On Fail $\Rightarrow p = \max(p_{min}, p/2)$
2) On Success $\Rightarrow p = \min(2p, p_{max})$

CSMA: (Carrier Sense Multiple Access)
- Check if someone is transmitting before sending

Utilization of TDMA = 1 if all nodes backlogged

Throughput of i^{th} TDMA node = $1/n$

" slotted ALOHA = $p(1-p)^{N-1}$

CSMA vs Stabalized ALOHA:
CSMA has ↑ utilization and more collisions. This is because CSMA nodes maintain high p's → so there are more sending attempts.

Carrier Sense: Allows nodes to check if others sending in current timeslot.

Contention window: Nodes guaranteed to attempt transmission in bounded time => more Fair.

TDMA packets removed from queue at rate $1/N$

TDMA Utilization with non uniform workload

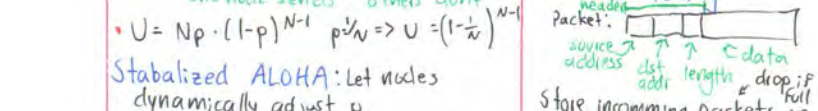
$U_{TDMA} = \frac{K}{N} + \frac{N\sum_{i=k+1}^{N} r_i}{N}$ (r_i: P node i sends)

(K: # first k nodes w growing queues $\Rightarrow U=1$; all others aren't backlogged all the time)

Switching

Circuit Switching:
- Guaranteed Rate
- Wasted link capacity if bursty
- Path established before sending
- All data follows 1 path
- No reordering, constant delay
- No dropped packages
- Can't route around Failures

Packet Switching



Store incomming packets in queue. Determine outgoing link of first packet using header.

Queues - Absorb bursts
Add queing delay and processing delay. Little's Law: $N = \lambda d$ (N: Avg # of packets in queue; λ: average rate of packet switch; d: Avg time packet spends on queue, mean delay per packet)

Rate at which packets arrive.

- No guarantees.
- More efficient
- Send data immediatly
- Different packages could follow diff paths
- Packets can be reordered, delayed or dropped
- Can route around Failures.

Total time on link = propagation time + transmission time

Total time = Propagation + trans + Queue + Processing

Routing

Distance Vector

"Talk to neighbors about everyone"

Ad: Each node sends:
List of all nodes node knows about + cost to only neighbors.

Integrate: For each (dst, cost)

Adj $y \to x$ IF x already using y to->dst: update cost

else if y can provide better path to dst: update routing

2L ads sent

- Use For very small loop-free networks
- Nodes don't know entire topology

Count to Infinity Problem

Path Vector Routing - Include min cost path in add. Integrate only if node not in path.
Use For large networks

Link State

"Talk to everyone about neighbors"

Ad: Each node sends
List of neighbors and cost to them.

To: neighbors who then forward to neighbors => All network "flood"

Integrate: Run dijkstra on complete map of network. ($O(N(N+E))$)

- Fast convergence => Out valid + Path visible
- can have routing loops but rare
- use For small networks

Route validity: $\forall n$ if routingT[dst]=k => There is a path from $N \to$ dst whose First hop is K

Path visibility: Every router w/ path to dst learns at lest 1 valid route to dst.

In Link State, nodes forward messages even if they are last in chain

Dijkstra:

```
W = all nodes; routingTable[self] = self
costTable[self] = 0  costTable[others] = ∞
While W not empty:
    v = node in W min cost to so far
    W.remove(v)
    for each neighbor v of u:
        d = costTable[u] + cost(u,v)
        if d < costTable[v]: # found shorter path
            costTable[v] = d
            routingTable[v] = routingT[u]
```

LSA Acds: 2 over each link? longest path in network

Time to notice failure

DV: $= 2 \cdot HELLO + ADVERT \cdot L_E$ (HELLO, ADVERT)

Reliable Transport

6.02 Quiz III

Stop and wait: Wait for ACK before sending again
Sender: - send packet. Remember seq num + time
- If ACK for that packet send next
- IF no Ack For Timeout secs => retransmit

Receiver: - Send ACK for received packet
- If packet is next one (check w/ seq num) send to app (Recognize duplicates)

Choosing Timeout: (since RTT changes)

$srtt[n] = \alpha \cdot rttSample + (1-\alpha) \cdot srtt[n-1]$

$devSample = |rttSample - srtt[n]|$

$srttdev[n] = \beta \cdot devSample + (1-\beta) \cdot srttdev[n-1]$

$timeout = srtt[n] + k \cdot srttdev[n]$

T = expected time b/w successfull delivery of pkts

$T = RTT + (L/(1-L)) \cdot timeout$

$L = 1-(1-p)^K$ (p: Probability that packet (or Ack) is lost; K: # hops in round trip; per link loss probability)

$T = RTT/(1-L)$

Throughput $= 1/T = (1-L)/RTT$

Problem: Bad utilization. Solution....

Sliding - Window

- Allow W un-ACKED packets
- window for outstanding packets does not necessarily contain succesive seq nums
- Sender can have max of W outstanding pkts.

Receiver: Keep packets in buffer until next expected packet arrives, then send as many as possible to app in order

Throughput $= W/RTT_{min}$ if $W \le B \cdot RTT_{min}$

$= B$ if $W > B \cdot RTT_{min}$ (B: bottleneck link)

Utilization at most $= 1-L$ (L: P that pkt or Ack is lost)

Setting $W = B \cdot RTT_{min} \Rightarrow$
Max throughput $= B$
or $(1-L)B$ w/ packet loss

$N = d/(1-L)$ (N: expected # of packets to send; L: P that packet or ack lost; d: # ACKs to receive)

$S = D_q \cdot PKT_{rate}$ (S: Avg size of queue; D_q: Queue delay (Little's); PKT_{rate}: packet rate)

$W_{max} = RTT_{min} \cdot \text{max throughput} + K$ (mean # of packets in queue; greater would experience packet loss)

$N = W \cdot RTT_{min} \cdot \frac{R}{L}$ (R: Bottleneck bits/second; L: L bits/packet; N: avg # packets in queue)

$D = WL/R$ (queueing delay)

With m users: $W_{max} = (RTT_{min} \cdot B + K)/m$ $N = mW \cdot RTT_{min}$ $D = mWL/R$