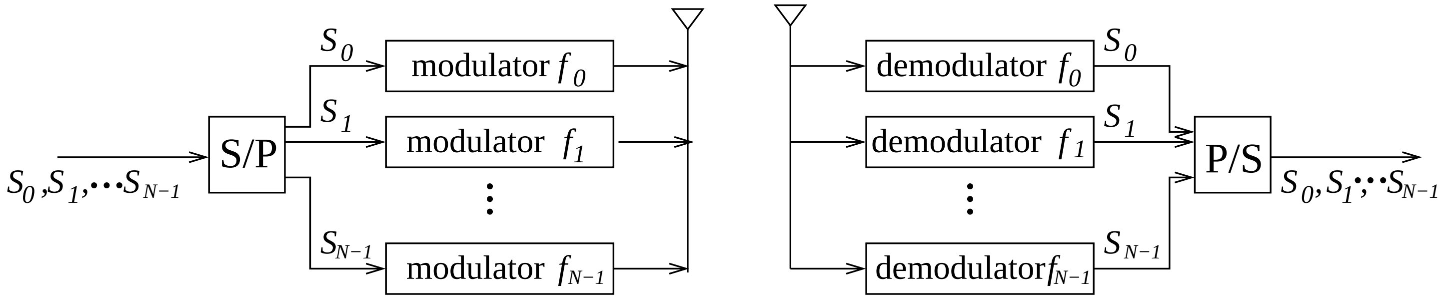
# Revision of Lecture Twenty

- Previous lecture focuses on interface between physical layer and network layer, referred to as **medium access control**
- Concepts of **user** and **signalling** (control) channels
- Random access (contention) algorithms
- This lecture we move back to physical layer, and look into **multicarrier** system

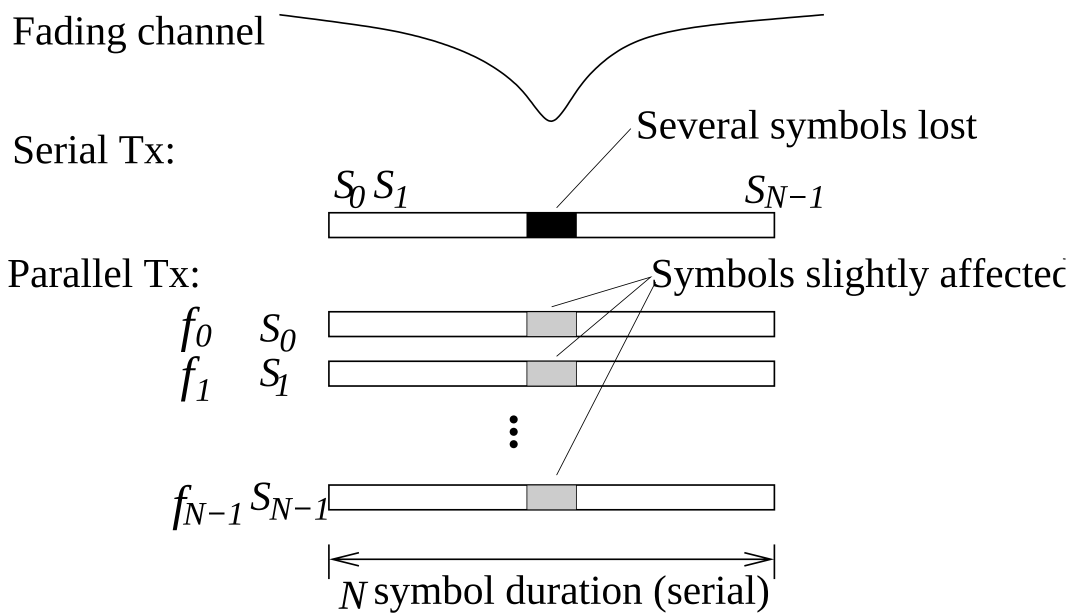
# Orthogonal Frequency Division Multiplexing

- OFDM applies **multicarrier** modulation principle by dividing the data stream into several bit streams, each of which has much lower bit rate, and using these substreams to modulate several carriers
- Basic **OFDM** system:



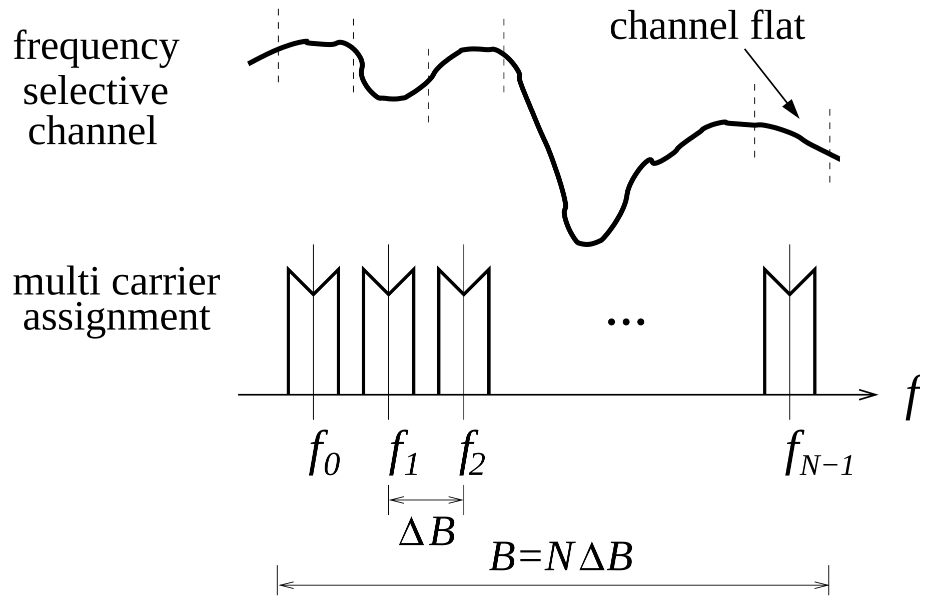

*What OFDM is good for?*

1. **Combating fading**: in a parallel transmission, each symbol in a sub-carrier has a much larger symbol duration, equal to $N$ times of the symbol duration in serial transmission. In a deep fade, several symbols in the single carrier system can be affected seriously and lost completely. However, in parallel transmission, each of the N symbols is only slightly affected and can still be recovered correctly



# OFDM (continue)

2. **Combating frequency selective**: the channel can be severely frequency selective, but for each sub-carrier, the sub-channel is flat or at least only slightly frequency selective




*What OFDM is bad for?*

- *High complexity*: to be effective, number of sub-carriers $N$ should be large

  If OFDM is implemented with $N$ modulators/demodulators, the complexity will be enormous.

  Fortunately, it can be implemented alternatively using DFT/FFT to reduce this high complexity

- Another disadvantage of OFDM systems is *high peak to average power*

  With $N$ sinusoidal signals added together, the peak amplitude becomes very large, which will be clicked by amplifier and channel's nonlinear saturation, causing distortion

# Fourier Transform Pair

- If a discrete-time aperiodic signal $x(k)$ satisfies $$\sum_{k=-\infty}^{\infty} |x(k)| < \infty$$

  then
  $$\text{FT:} \quad X(\omega) = \sum_{k=-\infty}^{\infty} x(k) \exp(-j\omega k) \qquad \text{IFT:} \quad x(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) \exp(j\omega k)\, d\omega$$

  Integration in IFT can also be over $0$ to $2\pi$
- Spectra: $X(\omega) = |X(\omega)| \exp(j\angle X(\omega))$, with $|X(\omega)|$ being the amplitude spectrum and $\angle X(\omega)$ the phase spectrum of $x(k)$
- Parseval's theorem:
  $$\sum_{k=-\infty}^{\infty} |x(k)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(\omega)|^2\, d\omega$$

  where $|X(\omega)|^2$ is the energy spectral density, giving distribution of signal energy in frequency domain. In practice, the power spectral density is more often used
- **Differences**:
  - Continuous-time: $f$ or $2\pi f$ has the unit of Hz or radian/s, and ranges in $(-\infty, \infty)$. FT is an integral
  - Discrete-time: $\omega$ has the unit of radian, and ranges in $[-\pi, \pi]$ or $[0, 2\pi]$. FT is a summation and $X(\omega)$ is periodic with period $2\pi$

# Discrete-Time Fourier Series

- If $x(k)$ is periodic with period $K$, i.e. $x(k) = x(k+K)$, $x(k)$ can be expressed by DFS:

$$x(k) = \sum_{n=0}^{K-1} c_n \exp(j\omega_n k), \quad \omega_n = \frac{2\pi n}{K}$$

  Note there are $K$ frequency components $\exp(j\omega_n k)$ for $0 \le n \le K-1$ and $0 \le \omega_n < 2\pi$, and the Fourier coefficients

$$c_n = \frac{1}{K} \sum_{k=0}^{K-1} x(k) \exp(-j\omega_n k), \quad 0 \le n \le K-1$$

  provide the amplitudes and phases for frequency components $\exp(j\omega_n k)$

- **Differences** in periodic signal:
  - Continuous-time: has infinite frequency components, and Fourier coefficients are integrals
  - Discrete-time: has finite frequency components, and Fourier coefficients are summations
- In theory, $X(\omega)$ is all we need but let us consider some practical constraints
  - Computing $X(\omega)$ requires infinite summation, that is, infinite number of samples $\longrightarrow$ one can only approximate it by a finite signal samples in a finite summation
  - Displaying $X(\omega)$ requires $\omega$ taking values continuously in $[0,\ 2\pi)$ $\longrightarrow$ one can only approximate it at finite discrete points $\omega_n$, that is, sample $X(\omega)$ and take only a finite spectrum samples.

  These considerations leads to discrete-time Fourier transform

# Discrete-Time Fourier Transform

- Windowing data so that $x(k) = 0$ for $k < 0$ and $k \geq L$, i.e. a finite sequence $x(k)$ of length $L$ $\longrightarrow$ the corresponding Fourier transform is

$$X(\omega) = \sum_{k=0}^{L-1} x(k) \exp(-j\omega k), \quad 0 \leq \omega < 2\pi$$

- Sample $X(\omega)$ at frequencies $\omega_n = 2\pi n/K$, $0 \leq n \leq K-1$, where $K \geq L$ $\longrightarrow$ the resulting spectrum samples or DFT of $\{x(k)\}$ is

$$X(n) = X(\omega_n) = \sum_{k=0}^{L-1} x(k) \exp(-j2\pi nk/K) = \sum_{k=0}^{K-1} x(k) \exp(-j2\pi nk/K)$$

- Inverse DFT (IDFT) is:

$$x(k) = \frac{1}{K} \sum_{n=0}^{K-1} X(n) \exp(j2\pi nk/K), \; 0 \leq k \leq K-1$$

- **DFT: time samples $\{x(k)\}$ of length $L \leq K$ $\Leftrightarrow$ frequency samples $\{X(n)\}$ of length $K$**
- For $K \geq L$, $\{x(k)\}_{k=0}^{L-1}$ can be exactly reconstructed from $\{X(n)\}_{n=0}^{K-1}$
  Otherwise, time folding or aliasing occurs $\longrightarrow$ This is dual to spectral folding or aliasing when sampling frequency is less than the Nyquist rate

# Example

For 6-point sequence $x(k) = k + 1$, $0 \le k \le 5$, the spectrum $X(\omega)$:

$$X(\omega) = \sum_{k=0}^{5} x(k) \exp(-j\omega k) = \sum_{k=0}^{5} (k+1) \exp(-j\omega k), \quad 0 \le \omega < 2\pi$$

Evaluate $X(\omega)$ at the 4 frequencies $\omega_n = 2\pi n/4$, $0 \le n \le 3$:

$$X(n) = \sum_{k=0}^{5} (k+1) \exp(-j2\pi nk/4), \quad 0 \le n \le 3$$

or

$$X(0) = 21, \; X(1) = 3 - 4j, \; X(2) = -3, \; X(3) = 3 + 4j$$

The IDFT for the resulting 4 samples $X(n)$, $0 \le n \le 3$:

$$\hat{x}(k) = \frac{1}{4} \sum_{n=0}^{3} X(n) \exp(j2\pi nk/4), \quad 0 \le k \le 3$$

or

$$\hat{x}(0) = 6, \; \hat{x}(1) = 8, \; \hat{x}(2) = 3, \; \hat{x}(3) = 4$$

This example illustrates time aliasing (note $x(0) = 1, x(1) = 2, x(2) = 3, x(3) = 4$)

To avoid time aliasing, frequency samples $K$ must be no less than time samples $L$

# Fast Fourier Transform

- Recall that DFT: $\{x(k)\}_{k=0}^{K-1} \Longleftrightarrow \{X(n)\}_{n=0}^{K-1}$. By introducing $W_K = \exp(-j2\pi/K)$,

$$\text{DFT:} \quad X(n) = \sum_{k=0}^{K-1} x(k) W_K^{kn}, \quad 0 \le n \le K-1$$

$$\text{IDFT:} \quad x(k) = \frac{1}{K} \sum_{n=0}^{K-1} X(n) W_K^{-kn}, \quad 0 \le k \le K-1$$

- Direct computation of DFT can be costly for large $K$: $2K^2$ trigonometric functions, $K^2$ multiplications, and $K(K-1)$ additions
- Let $K = LM$. Data can either be stored in one-dimensional array: $\{x(k)\}$ with $0 \le k \le K-1$ or in two-dimensional array: $x(l, m)$ indexed by $l$ and $m$ with $0 \le l \le L-1$ and $0 \le m \le M-1$
- Row wise:

$k = Ml + m$

| | | | | | |
|---|---|---|---|---|---|
| $x(0,0)$ | $\cdots$ | $x(0, M-1)$ | $x(0)$ | $\cdots$ | $x(M-1)$ |
| $x(1,0)$ | $\cdots$ | $x(1, M-1)$ | $x(M)$ | $\cdots$ | $x(2M-1)$ |
| | $\vdots$ | | | $\vdots$ | |
| $x(L-1,0)$ | $\cdots$ | $x(L-1, M-1)$ | $x((L-1)M)$ | $\cdots$ | $x(LM-1)$ |

- Column wise:

$k = l + mL$

| | | | | | |
|---|---|---|---|---|---|
| $x(0,0)$ | | $x(0, M-1)$ | $x(0)$ | | $x((M-1)L)$ |
| $x(1,0)$ | $\cdots$ | $x(1, M-1)$ | $x(1)$ | $\cdots$ | $x((M-1)L+1)$ |
| $\vdots$ | | $\vdots$ | $\vdots$ | | $\vdots$ |
| $x(L-1,0)$ | | $x(L-1, M-1)$ | $x(L-1)$ | | $x(LM-1)$ |

# FFT Algorithms

- Similarly, $X(n), 0 \le n \le K-1 \Longleftrightarrow X(p,q), 0 \le p \le L-1, 0 \le q \le M-1$ with row wise: $n = Mp + q$ or column wise: $n = p + qL$
- Assuming column wise for $x(k)$ and row wise for $X(n)$, then

$$X(p,q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x(l,m) W_K^{(l+mL)(Mp+q)}$$

where $W_K^{(l+mL)(Mp+q)} = W_K^{Mlp} W_K^{Kmp} W_K^{lq} W_K^{Lmq}$. But $W_K^{Mlp} = W_{K/M}^{lp} = W_L^{lp}$, $W_K^{Kmp} = 1$, and $W_K^{Lmq} = W_{K/L}^{mq} = W_M^{mq}$. Thus:

$$X(p,q) = \underbrace{\sum_{l=0}^{L-1} \underbrace{\left( W_K^{lq} \underbrace{\left[ \sum_{m=0}^{M-1} x(l,m) W_M^{mq} \right]}_{\text{step 1}} \right)}_{\text{step 2}} W_L^{lp}}_{\text{step 3}}$$

- The computation of DFT can be divided into three steps as shown in the next slide

# FFT Algorithms (continue)

- Algorithm one:
  1. For $0 \leq l \leq L-1$, compute the $M$-point DFTs:
  $$F(l,q) = \sum_{m=0}^{M-1} x(l,m) W_M^{mq}, \quad 0 \leq q \leq M-1$$
  2. For $0 \leq l \leq L-1$ and $0 \leq q \leq M-1$, compute the array $G(l,q) = W_K^{lq} F(l,q)$
  3. For $0 \leq q \leq M-1$, compute the $L$-point DFTs
  $$X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp}, \quad 0 \leq p \leq L-1$$
- Rearrange the double summation in the same DFT expression $\Rightarrow$ another similar algorithm
- Choosing row wise for $x(k)$ and column wise for $X(n)$ $\Rightarrow$ two more similar algorithms
- Complexity of these 4 algorithms resulting from a two-stage decomposition is: $2(L^2 + M^2 + K)$ trigonometric functions, $K(M+L+1)$ multiplications, $K(M+L-2)$ additions
- With $L = 2$ and $M = \frac{K}{2}$, for example, complexity reduction factor is approximately 2
- Factoring $K = r_1 r_2 \cdots r_v$, with $v$ the stage decomposition, leads to the computation of many small DFTs and, the more stage $v$, the more significant in complexity reduction

# Radix-2 FFT Algorithms

- When $K = r^v$, DFTs are of size $r$ and computation has regular pattern, where $r$ is called the radix of FFT algorithm. In particular, with $K = 2^v$, we have radix-2 FFT algorithms
- **Decimation-in-frequency** FFT: in the decomposition stage one, choose $L = K/2$ and $M = 2$:

$$X(n) = \sum_{k=0}^{K/2-1} x(k)W_K^{kn} + W_K^{nK/2} \sum_{k=0}^{K/2-1} x(k+K/2)W_K^{kn}$$

- Since $W_K^{nK/2} = (-1)^n$

$$X(n) = \sum_{k=0}^{K/2-1} \left(x(k) + (-1)^n x(k+K/2)\right) W_K^{kn}, \quad 0 \le n \le K-1$$

- Next decimate $X(n)$ into even and odd samples and use $W_K^2 = W_{K/2}$:

$$X(2n) = \sum_{k=0}^{K/2-1} \left(x(k) + x(k+K/2)\right) W_{K/2}^{kn} \quad n = 0, 1, \cdots, \frac{K}{2} - 1$$

$$X(2n+1) = \sum_{k=0}^{K/2-1} \left[\left(x(k) - x(k+K/2)\right) W_K^{k}\right] W_{K/2}^{kn} \quad n = 0, 1, \cdots, \frac{K}{2} - 1$$

# Radix-2 FFT Algorithms (continue)

- Define two $K/2$-point sequences

$$\left.\begin{aligned} g_1(k) &= x(k) + x(k+K/2) \\ g_2(k) &= \left[x(k) - x(k+K/2)\right] W_K^k \end{aligned}\right\} k = 0, 1, \cdots, \frac{K}{2} - 1$$

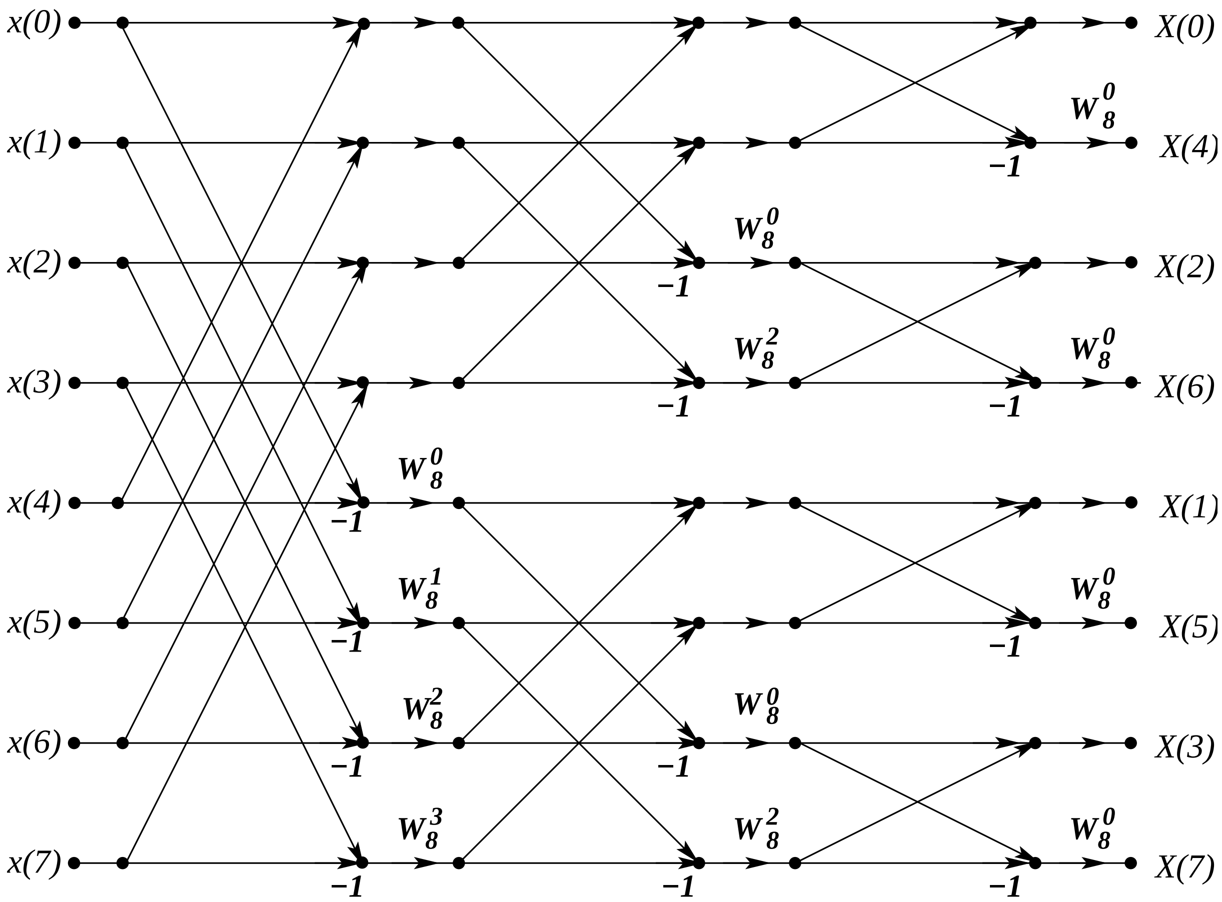
- Then

$$X(2n) = \sum_{k=0}^{K/2-1} g_1(k) W_{K/2}^{kn}, \quad X(2n+1) = \sum_{k=0}^{K/2-1} g_2(k) W_{K/2}^{kn}$$

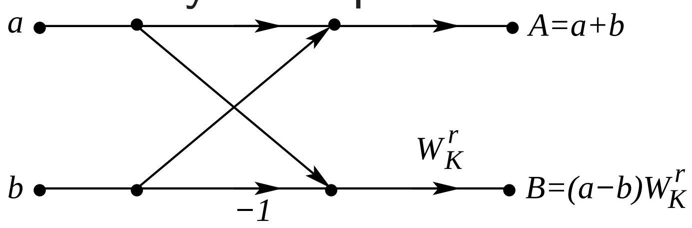
- $K/2$-point DFTs $X(2n)$ and $X(2n+1)$ can each be decimated into two $K/4$-point DFTs
- Procedure is repeated and entire procedure involves $v = \log_2(K)$ stages of decimation
- **Decimation-in-time** FFT: decimate $\{x(k)\}$ into even and odd samples and repeat the procedure
- Radix-2 FFT algorithm complexity: $(K/2)\log_2(K)$ complex multiplications, $K\log_2(K)$ complex additions
- **Example**. $1024$-point DFT with $K = 2^{10}$: direct computing involves 1048576 multiplications and 1047552 additions, but radix-2 FFT only involves 5120 multiplications and 10240 additions $\rightarrow$ speed improvement factor is approximately 100

# 8-Point Decimation-in-Frequency FFT

Algorithm:




Basic operation – "butterfly" computation



# Summary

- OFDM: basic concepts, effective in combating channel fading and frequency selective, and disadvantages
- Frequency analysis of discrete-time signals: differences with continuous-time case
- DFT: $\{x(k)\}_{k=0}^{K} \Longleftrightarrow \{X(n)\}_{n=0}^{K}$, practical considerations, time aliasing
- FFT: basic concepts, Radix-2, DFT implemented efficiently by FFT is widely used in communication systems