CS 414 – Multimedia Systems Design

# Lecture 6 – Basics of Compression (Part 1)

Klara Nahrstedt

Spring 2009

# Administrative

- MP1 is posted
- Discussion meeting today Monday, February 2, at 7pm, 3401 SC.

# Need for Compression

- Uncompressed audio
- 8 KHz, 8 bit
  - 8K per second
  - 30M per hour
- 44.1 KHz, 16 bit
  - 88.2K per second
  - 317.5M per hour
- 100 Gbyte disk holds 315 hours of CD quality music

- Uncompressed video
- 640 x 480 resolution, 8 bit color, 24 fps
  - 7.37 Mbytes per second
  - 26.5 Gbytes per hour
- 640 x 480 resolution, 24 bit (3 bytes) color, 30 fps
  - 27.6 Mbytes per second
  - 99.5 Gbytes per hour
- 100 Gbyte disk holds 1 hour of high quality video

# Broad Classification

- Entropy Coding (statistical)
  - lossless; independent of data characteristics
  - e.g. RLE, Huffman, LZW, Arithmetic coding
- Source Coding
  - lossy; may consider semantics of the data
  - depends on characteristics of the data
  - e.g. DCT, DPCM, ADPCM, color model transform
- Hybrid Coding (used by most multimedia systems)
  - combine entropy with source encoding
  - e.g., JPEG-2000, H.264, MPEG-2, MPEG-4, MPEG-7

# Data Compression

- Branch of information theory
  - minimize amount of information to be transmitted
- Transform a sequence of characters into a new string of bits
  - same information content
  - length as short as possible

# Concepts

- Coding (the code) maps source messages from alphabet (*A*) into code words (*B*)
- Source message (symbol) is basic unit into which a string is partitioned
  - can be a single letter or a string of letters
- EXAMPLE: aa bbb cccc ddddd eeeeee fffffffgggggggg
  - *A* = {a, b, c, d, e, f, g, space}
  - *B* = {0, 1}

# Taxonomy of Codes

- Block-block
  - source msgs and code words of fixed length; e.g., ASCII
- Block-variable
  - source message fixed, code words variable; e.g., Huffman coding
- Variable-block
  - source variable, code word fixed; e.g., RLE, LZW
- Variable-variable
  - source variable, code words variable; e.g., Arithmetic

# Example of Block-Block

- Coding “aa bbb cccc ddddd eeeeee fffffffgggggggg”
- Requires 120 bits

| Symbol | Code word |
|---|---|
| a | 000 |
| b | 001 |
| c | 010 |
| d | 011 |
| e | 100 |
| f | 101 |
| g | 110 |
| space | 111 |

# Example of Variable-Variable

- Coding “aa bbb cccc ddddd eeeeee fffffffgggggggg”

- Requires 30 bits
  - don’t forget the spaces

| Symbol | Code word |
|---|---|
| aa | 0 |
| bbb | 1 |
| cccc | 10 |
| ddddd | 11 |
| eeeeee | 100 |
| fffffff | 101 |
| gggggggg | 110 |
| space | 111 |

# Concepts (cont.)

- A code is
  - *distinct* if each code word can be distinguished from every other (mapping is one-to-one)
  - *uniquely decodable* if every code word is identifiable when immersed in a sequence of code words
    - e.g., with previous table, message 11 could be defined as either ddddd or bbbbbb

# Static Codes

- Mapping is fixed before transmission
  - message represented by same codeword every time it appears in message (ensemble)
  - Huffman coding is an example
- Better for independent sequences
  - probabilities of symbol occurrences must be known in advance;

# Dynamic Codes

- Mapping changes over time
  - also referred to as *adaptive* coding
- Attempts to exploit locality of reference
  - periodic, frequent occurrences of messages
  - dynamic Huffman is an example
- Hybrids?
  - build set of codes, select based on input

# Traditional Evaluation Criteria

- Algorithm complexity
  - running time
- Amount of compression
  - redundancy
  - compression ratio
- How to measure?

# Measure of Information

- Consider symbols $s_i$ and the probability of occurrence of each symbol $p(s_i)$
- In case of **fixed-length coding** , smallest number of bits per symbol needed is
  - $L \geq log_2(N)$ bits per symbol
  - Example: Message with 5 symbols need 3 bits $(L \geq log_2 5)$

# Variable-Length Coding-Entropy

- What is the minimum number of bits per symbol?
- Answer: Shannon's result – theoretical minimum average number of bits per code work is known as Entropy (H)

$$\sum_{i=1}^{n} -p(s_i)\log_2 p(s_i)$$

# Entropy Example

- Alphabet = {A, B}
  - p(A) = 0.4; p(B) = 0.6
- Compute Entropy (H)
  - $-0.4*\log_2 0.4 + -0.6*\log_2 0.6 = .97$ bits

# Compression Ratio

- Compare the average message length and the average codeword length
  - e.g., average L(message) / average L(codeword)

- Example:
  - {aa, bbb, cccc, ddddd, eeeeee, fffffff, gggggggg}
  - Average message length is 5
  - If we use code-words from slide 9, then
    - We have {0,1,10,11,100,101,110}
    - Average codeword length is 2.14.. Bits
  - Compression ratio: 5/2.14 = 2.336

# Symmetry

- Symmetric compression
  - requires same time for encoding and decoding
  - used for live mode applications (teleconference)
- Asymmetric compression
  - performed once when enough time is available
  - decompression performed frequently, must be fast
  - used for retrieval mode applications (e.g., an interactive CD-ROM)

# Entropy Coding Algorithms (Content Dependent Coding)

- Run-length Encoding (RLE)
  - Replaces sequence of the same consecutive bytes with number of occurrences
  - Number of occurrences is indicated by a special flag (e.g., !)
  - Example:
    - abcccccccccdeffffggg  (20 Bytes)
    - abc!9def!4ggg (13 bytes)

# Variations of RLE (Zero-suppression technique)

- Assumes that only one symbol appears often (blank)
- Replace blank sequence by M-byte and a byte with number of blanks in sequence
  - Example: M3, M4, M14,…
- Some other definitions are possible
  - Example:
    - M4 = 8  blanks, M5 = 16 blanks, M4M5=24 blanks

# Huffman Encoding

- Statistical encoding
- To determine Huffman code, it is useful to construct a binary tree
- Leaves are characters to be encoded
- Nodes carry occurrence probabilities of the characters belonging to the subtree

- Example: How does a Huffman code look like for symbols with statistical symbol occurrence probabilities:

  P(A) = 8/20, P(B) = 3/20, P(C ) = 7/20, P(D) = 2/20?

# Huffman Encoding (Example)

**Step 1 : Sort all Symbols according to their probabilities (left to right) from Smallest to largest**

**these are the leaves of the Huffman tree**

P(B) = 0.51

P(C) = 0.09

P(E) = 0.11

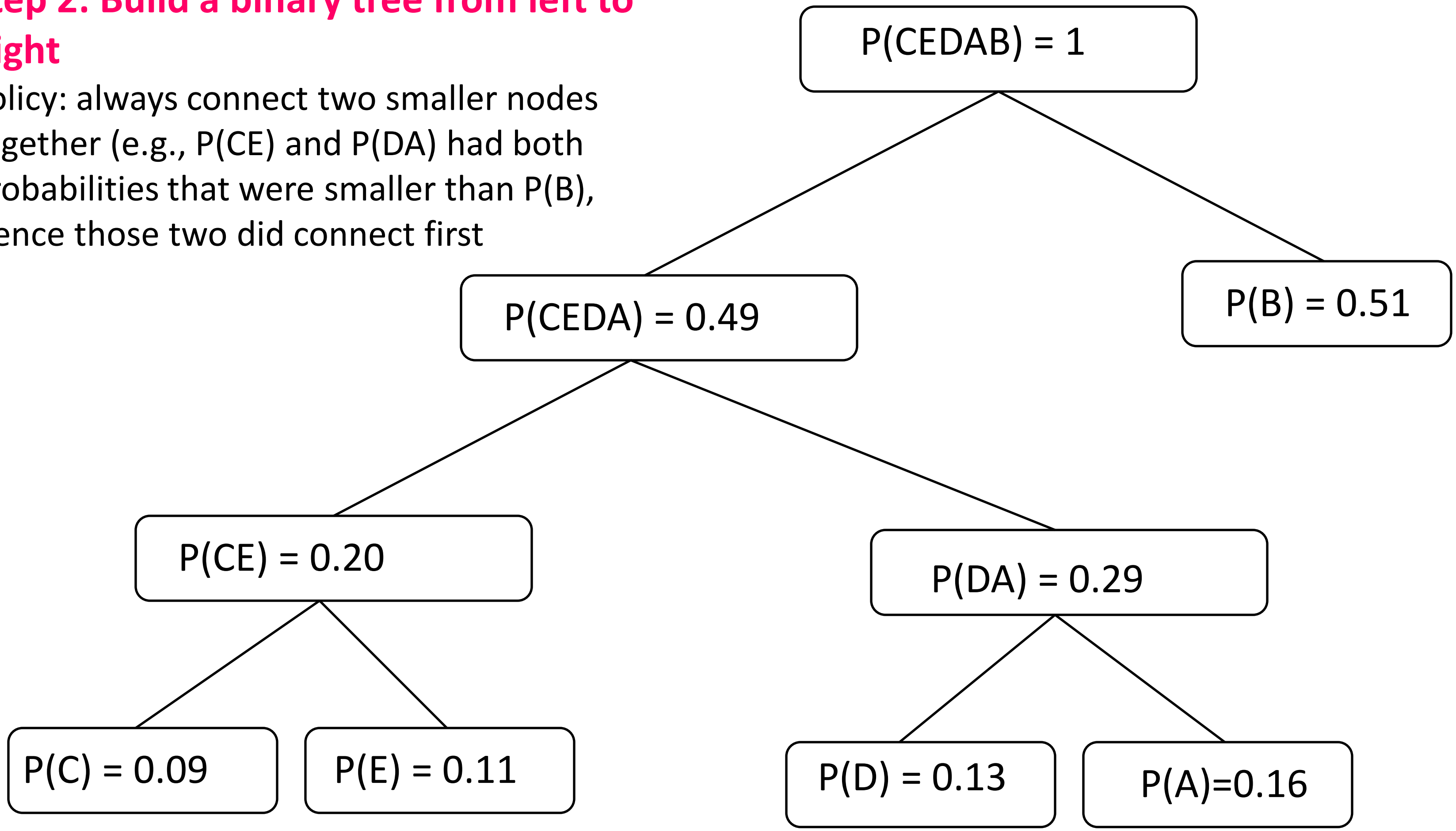
P(D) = 0.13

P(A)=0.16

# Huffman Encoding (Example)

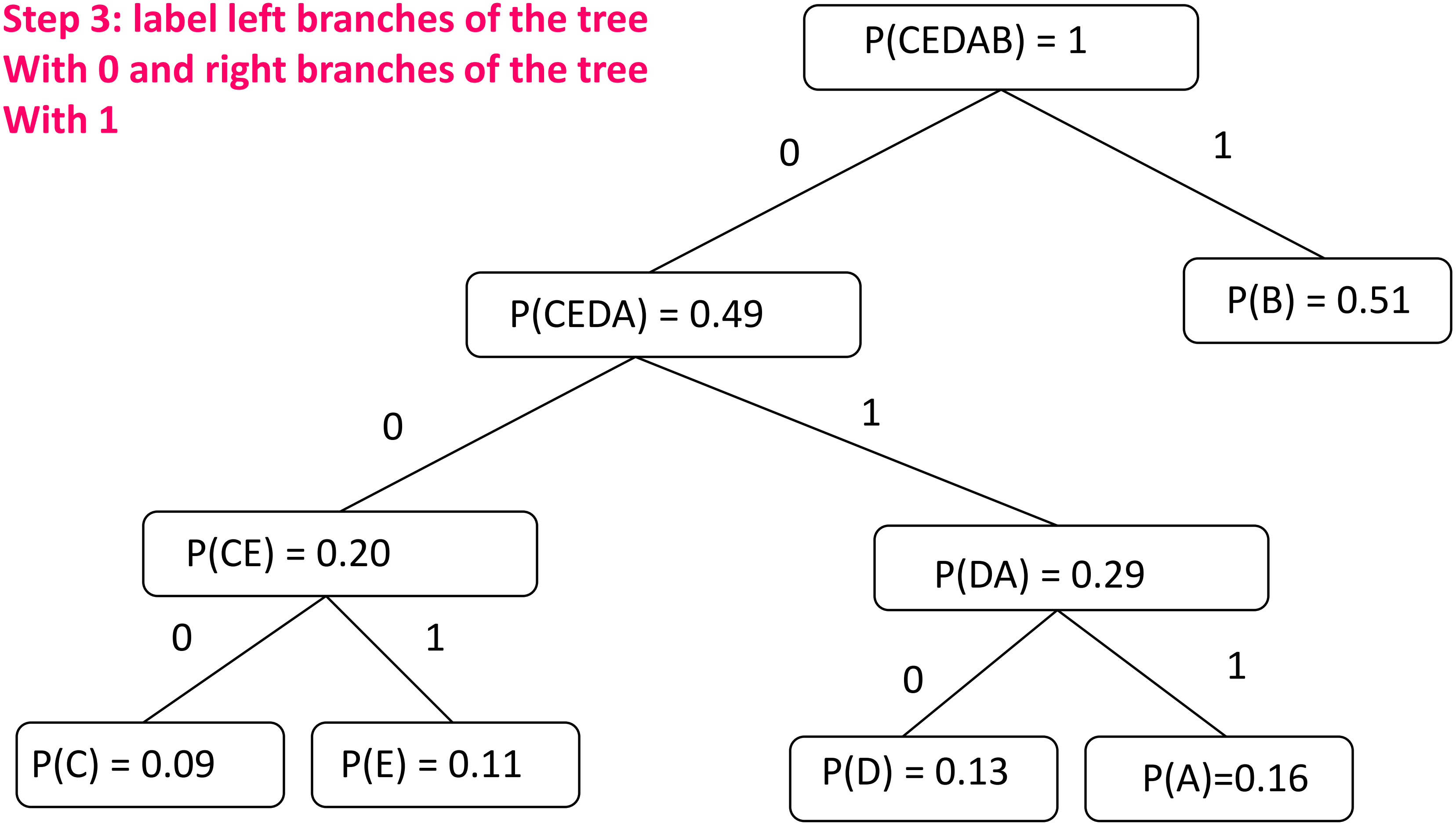
**Step 2: Build a binary tree from left to Right**

Policy: always connect two smaller nodes together (e.g., P(CE) and P(DA) had both Probabilities that were smaller than P(B), Hence those two did connect first



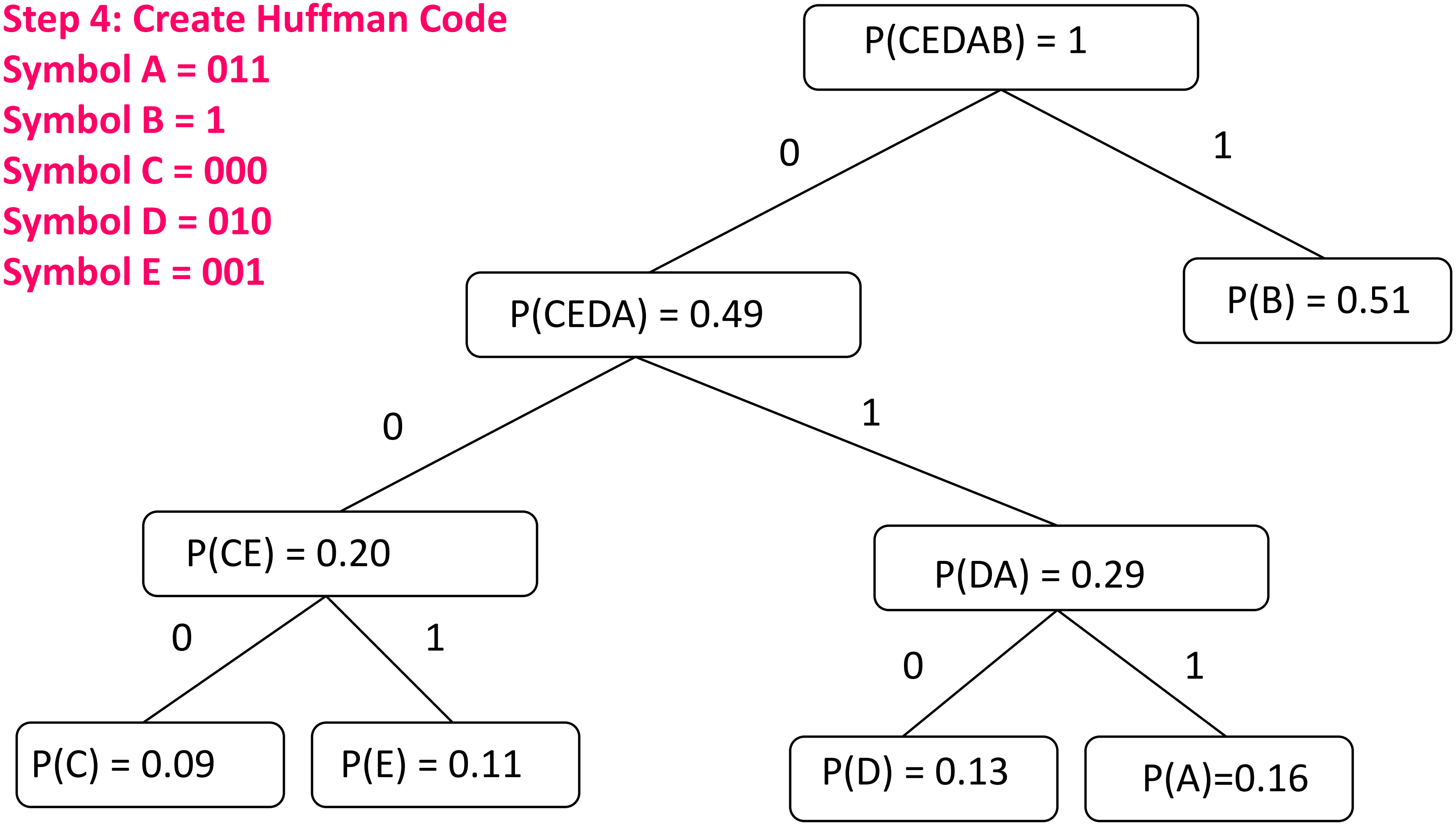
# Huffman Encoding (Example)

**Step 3: label left branches of the tree With 0 and right branches of the tree With 1**



# Huffman Encoding (Example)

**Step 4: Create Huffman Code**
**Symbol A = 011**
**Symbol B = 1**
**Symbol C = 000**
**Symbol D = 010**
**Symbol E = 001**



# Summary

- Compression algorithms are of great importance when processing and transmitting
  - Audio
  - Images
  - Video