Môn : Phát triển hệ thống đa phương tiện DHCNTT18ABTT

Ôn tập GK DPT

Hồ Phúc Lâm

Câu 1:

a)Cho Code Point U+00E6, U+00FC. Hãy xác định mã UTF-8 2 bytes tương ứng.

1.1.a. Cho Code Point U+00E6

Bước 1: Viết Code Point dưới dạng chuỗi nhị phân

U+00E6 => 00E6 => $0000\ 0000\ 1110\ 0110_{(2)}$

Vì cần chuyển đổi sang dạng UTF-8 2 bytes nên:

Lấy 11 bits từ bên phải sang bên trái

0000 0000 1110 0110 => 000 1110 0110

Tiếp theo, chia thành 2 phần gồm I 5 bits cao xét bên trái và II 6 bits thấp xét bên phải

000 1110 0110

Bước 2: đưa xuống bảng

1	1	0	0	0	0	1	1
1	0	1	0	0	1	1	0

Bước 3: đáp số C3 A6

C	3
A	6

Đối chiếu kết quả

Unicode code point	**character**	**UTF-8 (hex.)**	**name**
U+00E6	æ	c3 a6	LATIN SMALL LETTER AE

1.2.a. Cho Code Point U+00FC

Bước 1: Viết Code Point dưới dạng chuỗi nhị phân

U+00FC => 00FC => $0000\ 0000\ 1111\ 1100_{(2)}$

Vì cần chuyển đổi sang dạng UTF-8 2 bytes nên:

Lấy 11 bits từ bên phải sang bên trái

0000 0000 1111 1100 => 000 1111 1100

Tiếp theo, chia thành 2 phần gồm I 5 bits cao xét bên trái và II 6 bits thấp xét bên phải

000 1111 1100

Bước 2: đưa xuống bảng

1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	0

Bước 3: đáp số

C	3
B	C

C3 BC

Đối chiếu kết quả

Unicode code point	**character**	**UTF-8 (hex.)**	**name**
U+00FC	ü	c3 bc	LATIN SMALL LETTER U WITH DIAERESIS

b) Mã utf-8 của "Đ" là 0xc4,0x90; "Ạ" là 0xe1,0xba,0xa0; "Ọ" là 0xe1,0xbb,0x8c; "Ô" là 0xc3,0x94. "Ệ" là 0xe1,0xbb,0x86. A/C hãy viết code Python giải mã để cho ra kết quả là "ĐẠI HỌC CÔNG NGHIỆP"

```
#Khai bao các chuỗi "Đại học công nghiệp" và thay thế các kí tự đặc biệt

#Thành các mã hóa UTF-8

text= b"\xc4\x90\xe1\xba\xa0I H\xe1\xbb\x8cC C\xc3\x94NG NGHI\xe1\xbb\x86P"

text_decode= text.decode('utf-8')

print(text_decode)
```

Câu 2:

a) Việc chuyển đổi từ tín hiệu Analog sang Digital với mục đích gì ?Trình bày quá trình chuyển đổi tín hiệu từ Analog sang Digital.

- **Chuyển đổi tín hiệu Analog** (tín hiệu tương tự, liên tục) sang tín hiệu Digital. Vì máy tính xử lý thông tin tin rời rạc (gồm 0 và 1), cho nên cần phải chuyển tín hiệu liên tục Analog sang tín hiệu số Digital để có thể lưu trữ và xử lý trên máy tính.
- Giúp lưu trữ và truyền tải tín hiệu. Dễ dàng xử lý và phân tích bằng thuật toán. Ít bị nhiễu và suy giảm hơn khi truyền qua các kênh khác nhau. Bảo toàn dữ liệu và sao lưu, phục hồi tín hiệu gốc.

Để chuyển đổi tín hiệu Analog sang Digital, người ta thực hiện qua 3 giai đoạn: Sampling, Quantization và Coding

- **Sampling (lấy mẫu):** Tín hiệu Analog thực tế (như nhiệt độ, độ ẩm, âm thanh,..) được lấy mẫu theo các khoảng thời gian nhất định. Mỗi khoảng thời gian, tín hiệu âm thanh sẽ được đo và lưu trữ giá trị của nó
- **Quantization (lượng tử hóa):** Các giá trị mẫu âm thanh được chuyển đổi thành các trạng thái 0 và 1. Ví dụ, tín hiệu có giá trị mẫu thấp hơn ngưỡng được mã hóa thành 0, trong khi tín hiệu có giá trị mẫu cao hơn ngưỡng được mã hóa thành 1.
- **Coding (mã hóa):** Tín hiệu số được biểu diễn bằng các bit. Quá trình này cho phép tín hiệu âm thanh được lưu trữ, truyền và tái tại lại một cách chính xác.

b)Viết chương trình đọc file Audio và hiển thị thông tin về file.

```python
# Import necessary library
from mutagen.mp3 import MP3

# Function to display MP3 properties
def show_mp3_properties(file_path):
    try:
        # Load the MP3 file
        audio = MP3(file_path)
        # Get properties
        duration = audio.info.length  # Duration in seconds
        bitrate = audio.info.bitrate / 1000  # Bitrate in kbps
        sample_rate = audio.info.sample_rate  # Sample rate in Hz
        channels = audio.info.channels  # Number of channels
        # Print properties
        print(f"Tập tin File: {file_path}")
        print(f"Độ dài Duration: {duration:.2f} seconds")
        print(f"Tốc độ truyền Bitrate: {bitrate} kbps")
        print(f"Mẫu Sample Rate: {sample_rate} Hz")
        print(f"Kênh Channels: {channels}")
    except Exception as e:
        print(f"Đã có lỗi xảy ra: {e}")

# Main block
if __name__ == "__main__":
    # Input: File path from user
    file_path = input("Nhập đường dẫn tới tập tin âm thanh Audio: ")
    # Run function to display MP3 properties
    show_mp3_properties(file_path)
```

c)Viết chương trình tạo tín hiệu Audio hình Sin hoặc Cosin với các tham số liên quan như tần số lấy mẫu, tần số tín hiệu, biên độ (các tham số này tự chọn)

```
# Import các thư viện
import numpy as np
import wave
import struct
import matplotlib.pyplot as plt

# Khai báo các biến cần sử dụng
frequency = 1000  # Tần số tín hiệu (Hz)
num_samples = 48000  # Số lượng mẫu
sampling_rate = 48000  # Tần số lấy mẫu (samples/second)
amplitude = 16000  # Biên độ tín hiệu
file = "audioCosineWave.wav"  # Tên tệp WAV

# Phương trình sóng Cosine
cosine_wave = [np.cos(2 * np.pi * frequency * x / sampling_rate) for x in
range(num_samples)]

#Phương trình sóng sin_wave
sine_wave = [np.sin(2*np.pi*frequency*x/sampling_rate) for x in range(num_samples)]

# Thiết lập các khung frames hoặc mẫu
nframes = num_samples
comptype = "NONE"
compname = "not compressed"
nchannels = 1
sampwidth = 2  # 16 bit

# Mở tệp WAV
wav_file = wave.open(file, "w")
wav_file.setparams((nchannels, sampwidth, int(sampling_rate), nframes, comptype,
compname))

# Thiết lập thông số cho tệp
for s in cosine_wave:
    wav_file.writeframes(struct.pack('h', int(s * amplitude)))

# Đóng tệp WAV
wav_file.close()

# Vẽ biểu đồ dữ liệu cho 300 mẫu đầu tiên
plt.plot(cosine_wave[:300])
plt.title("Biểu đồ sóng Cos")
plt.xlabel("Mẫu (Samples)")
plt.ylabel("Biên độ (Amplitude)")
plt.grid(True)
plt.show()
```

```
# Import các thư viện
import numpy as np;import wave;import struct;import matplotlib.pyplot as plt
# Hàm tạo sóng âm thanh Sin hoặc Cos
def create_wave(wave_type, frequency, sampling_rate, num_samples, amplitude, file_name):
    if wave_type == "sin":
        # Phương trình sóng Sin
        wave_data = [np.sin(2 * np.pi * frequency * x / sampling_rate) for x in
range(num_samples)]
    elif wave_type == "cos":
        # Phương trình sóng Cos
        wave_data = [np.cos(2 * np.pi * frequency * x / sampling_rate) for x in
range(num_samples)]
    else:
        print("Loại sóng không hợp lệ! Vui lòng chọn 'sin' hoặc 'cos'.")
        return
    # Thiết lập các khung frames hoặc mẫu
    nframes = num_samples
    comptype = "NONE"
    compname = "not compressed"
    nchannels = 1
    sampwidth = 2  # 16 bit
    # Mở tệp WAV
    wav_file = wave.open(file_name, "w")
    wav_file.setparams((nchannels, sampwidth, int(sampling_rate), nframes, comptype,
compname))
    # Thiết lập thông số cho tệp
    for s in wave_data:
        wav_file.writeframes(struct.pack('h', int(s * amplitude)))
    # Đóng tệp WAV
    wav_file.close()
    # Xác định số lượng mẫu nhỏ để hiển thị, ví dụ lấy 1% tổng số mẫu hoặc ít nhất 100 mẫu
    display_samples = max(int(num_samples * 0.01), 100)
    # Vẽ biểu đồ dữ liệu cho phần mẫu nhỏ
    plt.plot(wave_data[:display_samples])
    plt.title(f"Biểu đồ sóng {wave_type.capitalize()}")
    plt.xlabel("Mẫu (Samples)")
    plt.ylabel("Biên độ (Amplitude)")
    plt.grid(True)
    plt.show()
# Hàm nhập các thông số từ người dùng
def main():
    # Nhập thông số từ người dùng
    wave_type = input("Chọn loại sóng (sin hoặc cos): ").lower()
    frequency = float(input("Nhập tần số tín hiệu (Hz): "))
    sampling_rate = int(input("Nhập tần số lấy mẫu (samples per second): "))
    num_samples = int(input("Nhập số lượng mẫu (samples): "))
    amplitude = int(input("Nhập biên độ tín hiệu: "))
    file_name = input("Nhập tên tệp WAV (ví dụ: audioWave.wav): ")
    # Gọi hàm để tạo sóng
    create_wave(wave_type, frequency, sampling_rate, num_samples, amplitude, file_name)
# Chạy chương trình
if __name__ == "__main__":
    main()
```

Câu 3:

a)A/C trình bày các mô hình màu RGB, YIQ, YUV và YCrCb.

- Mô hình màu RGB: Mỗi màu hiển thị được mô tả gồm các tham số độc lập – độ chói của các tham số chính được sử dụng trong màn hình CRT màu. Là mô hình màu được kết hợp bởi 3 thành phần cơ bản là Red – Green – Blue (RGB) để tạo ra các màu sắc khác nhau. Và có giá trị từ 0 đến 255 mỗi giá trị màu.
- Mô hình màu YIQ (Luminance, In-phase, Quadrature): là mô hình màu được sử dụng trong truyền hình Analog của Mỹ, đặc biệt là trong hệ thống truyền hình NTSC. Y đại diện cho độ sáng (luminance) của hình ảnh (thành phần này tương đương với đen trắng trong hệ thống truyền hình cũ). I và Q là hai thành phần màu chịu trách nhiệm truyền tải thông tin về sắc độ (chrominance), với I (in-phase): thành phần chênh lệnh pha. Q (quadurature): thành phần vuông pha.
- Mô hình màu YUV (Lumninace, Chrominace): là mô hình màu được sử dụng trong truyền hình màu, đặc biệt là hệ thống PAL và SECAM ở Châu Âu và một số nơi khác. Y đại diện cho độ sáng (Lumninace), tương tự như YIQ. U và V là hai thành phần màu sắc (chrominace) xác định sự khác biệt giữa màu xanh lam và độ sang (U) cũng như sự khác biệt giữa màu đỏ và độ sáng (V).
- Mô hình màu YcrCb (Luminace, Chrominace Blue, Chromiance Red) là phiên bản số hóa của YUV, được sử dụng chủ yếu trong các hệ thống xử lý video kỹ thuật số, bao gồm nén hình ảnh (JPEG) và nén video (MPEG). Y đại diện cho độ sáng (luminace). Cr (Chrominace Red) đại diện cho sự khác biệt giữa màu đỏ và độ sáng. Cb (chrominace Blue) đại diện cho sự khác biệt giữa màu xanh lam và độ sáng.

b)Dữ liệu video có tốc độ truyền 24 fps, mỗi frame có kích thước 720x486, mô hình YCrCb chroma subsampling sử dụng là 4:2:2.Xác định dung lượng lưu trữ video trong 1 phút 15 giây.

Vì là mô hình YcrCb chroma subampling sử dụng là 4:2:2, cho nên mỗi điểm ảnh pixel của ảnh sẽ sử dụng 2 bytes.

Vậy nên:

Tốc độ truyền Bit-rate = H x W x Dept x FPS = 720 x 486 x 2 bytes x 24 = 16 796 160

Dung lượng lưu trữ video trong 1 phút 15 giây (~75 giây)

C = Bit-rate x thời gian t = 16796160 x 75 = 1 259 712 000 bytes = 1201.3 MB

Đáp số: 1 259 712 000 Bytes = 1201.3 MB

Câu 4:

a)Viết chương trình hiển thị nội dung ảnh có định dạng jpg, png, bitmap

```
# Import thư viện Pillow
from PIL import Image

# Hàm hiển thị nội dung ảnh
def display_image(image_path):
    try:
        # Mở ảnh
        image = Image.open(image_path)
        # Hiển thị ảnh
        image.show()
        print(f"Hiển thị ảnh: {image_path}")
    except Exception as e:
        print(f"Không thể mở ảnh. Lỗi: {e}")

# Gọi hàm và truyền vào đường dẫn đến ảnh (jpg, png, bmp)
image_path = input("Nhập đường dẫn đến tệp ảnh (jpg, png, bmp): ")
display_image(image_path)
```

b)Viết chương trình thực hiện các chức năng biến đổi ảnh như : làm mờ ảnh

```
# Import thư viện Pillow
from PIL import Image, ImageFilter

# Hàm làm mờ ảnh
def blur_image(image_path, output_path):
    try:
        # Mở ảnh
        image = Image.open(image_path)
        # Làm mờ ảnh
        blurred_image = image.filter(ImageFilter.BLUR)
        # Lưu ảnh làm mờ
        blurred_image.save(output_path)
        # Hiển thị ảnh làm mờ
        blurred_image.show()
        print(f"Ảnh làm mờ đã được lưu tại: {output_path}")
    except Exception as e:
        print(f"Không thể làm mờ ảnh. Lỗi: {e}")

# Nhập đường dẫn ảnh và tên file đầu ra từ người dùng
image_path = input("Nhập đường dẫn đến tệp ảnh (jpg, png, bmp): ")
output_path = input("Nhập tên tệp cho ảnh làm mờ (ví dụ: blurred_image.png): ")

# Gọi hàm làm mờ ảnh
blur_image(image_path, output_path)
```

c) Viết chương trình thực hiện các chức năng biến đổi ảnh như xoay ảnh theo 1 góc tùy ý (góc <=20 độ)

```
# Import thư viện Pillow
from PIL import Image

# Hàm xoay ảnh
def rotate_image(image_path, output_path, angle):
    try:
        # Mở ảnh
        image = Image.open(image_path)
        # Xoay ảnh theo góc
        rotated_image = image.rotate(angle)
        # Lưu ảnh đã xoay
        rotated_image.save(output_path)
        # Hiển thị ảnh đã xoay
        rotated_image.show()
        print(f"Ảnh đã được xoay và lưu tại: {output_path}")
    except Exception as e:
        print(f"Không thể xoay ảnh. Lỗi: {e}")

# Hàm chính
def main():
    # Nhập đường dẫn ảnh
    image_path = input("Nhập đường dẫn đến tệp ảnh (jpg, png, bmp): ")
    # Nhập góc xoay
    angle = float(input("Nhập góc xoay (<= 20 độ): "))

    # Kiểm tra góc xoay
    if abs(angle) > 20:
        print("Góc xoay không hợp lệ! Vui lòng nhập góc <= 20 độ.")
        return

    # Nhập tên tệp cho ảnh đã xoay
    output_path = input("Nhập tên tệp cho ảnh đã xoay (ví dụ: rotated_image.png): ")

    # Gọi hàm để xoay ảnh
    rotate_image(image_path, output_path, angle)

# Chạy chương trình
if __name__ == "__main__":
    main()
```

d) Viết chương trình thực hiện các chức năng biến đổi ảnh như co giãn ảnh.

```
# Import thư viện Pillow
from PIL import Image

# Hàm co giãn ảnh
def resize_image(image_path, output_path, scale_width, scale_height):
    try:
        # Mở ảnh
        image = Image.open(image_path)

        # Tính toán kích thước mới
        new_size = (int(image.width * scale_width), int(image.height * scale_height))

        # Co giãn ảnh
        resized_image = image.resize(new_size)

        # Lưu ảnh đã co giãn
        resized_image.save(output_path)

        # Hiển thị ảnh đã co giãn
        resized_image.show()
        print(f"Ảnh đã được co giãn và lưu tại: {output_path}")
    except Exception as e:
        print(f"Không thể co giãn ảnh. Lỗi: {e}")

# Hàm chính
def main():
    # Nhập đường dẫn ảnh
    image_path = input("Nhập đường dẫn đến tệp ảnh (jpg, png, bmp): ")

    # Nhập tỷ lệ co giãn cho chiều rộng và chiều cao
    scale_width = float(input("Nhập tỷ lệ co giãn cho chiều rộng (0 < scale_width <= 1 để
thu nhỏ, > 1 để phóng to): "))
    scale_height = float(input("Nhập tỷ lệ co giãn cho chiều cao (0 < scale_height <= 1 để
thu nhỏ, > 1 để phóng to): "))

    # Nhập tên tệp cho ảnh đã co giãn
    output_path = input("Nhập tên tệp cho ảnh đã co giãn (ví dụ: resized_image.png): ")

    # Gọi hàm để co giãn ảnh
    resize_image(image_path, output_path, scale_width, scale_height)

# Chạy chương trình
if __name__ == "__main__":
    main()
```

e) Viết chương trình thực hiện các chức năng biến đổi ảnh như lật ảnh từ trái sang phải(hoặc ngược lại)

f) Viết chương trình thực hiện các chức năng biến đổi ảnh như lật ảnh từ trên xuống dưới(hoặc ngược lại).

```
# Import thư viện Pillow
from PIL import Image

# Hàm lật ảnh
def flip_image(image_path, output_path, direction):
    try:
        # Mở ảnh
        image = Image.open(image_path)

        # Lật ảnh theo hướng
        if direction == 'h':  # Lật ngang (trái sang phải)
            flipped_image = image.transpose(Image.FLIP_LEFT_RIGHT)
        elif direction == 'v':  # Lật dọc (trên xuống dưới)
            flipped_image = image.transpose(Image.FLIP_TOP_BOTTOM)
        else:
            print("Hướng lật không hợp lệ. Vui lòng nhập 'h' để lật ngang hoặc 'v' để lật
dọc.")
            return

        # Lưu ảnh đã lật
        flipped_image.save(output_path)

        # Hiển thị ảnh đã lật
        flipped_image.show()
        print(f"Ảnh đã được lật và lưu tại: {output_path}")
    except Exception as e:
        print(f"Không thể lật ảnh. Lỗi: {e}")

# Hàm chính
def main():
    # Nhập đường dẫn ảnh
    image_path = input("Nhập đường dẫn đến tệp ảnh (jpg, png, bmp): ")

    # Nhập hướng lật ('h' cho lật ngang, 'v' cho lật dọc)
    direction = input("Nhập hướng lật (h cho lật ngang, v cho lật dọc): ").lower()

    # Nhập tên tệp cho ảnh đã lật
    output_path = input("Nhập tên tệp cho ảnh đã lật (ví dụ: flipped_image.png): ")

    # Gọi hàm để lật ảnh
    flip_image(image_path, output_path, direction)

# Chạy chương trình
if __name__ == "__main__":
    main()
```

Câu 5: Viết chương trình phát hiện khuôn mặt người trên ảnh.

```
import cv2

def detect_faces(image_path):
    # Tải mô hình phát hiện khuôn mặt (Haar Cascade)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

    # Đọc ảnh
    image = cv2.imread(image_path)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  # Chuyển sang ảnh xám

    # Phát hiện khuôn mặt
    faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5)

    # Vẽ hình chữ nhật quanh khuôn mặt
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)  # Màu đỏ

    # Hiển thị ảnh với khuôn mặt đã phát hiện
    cv2.imshow('Detected Faces', image)
    cv2.waitKey(0)  # Chờ nhấn phím để đóng cửa sổ
    cv2.destroyAllWindows()

# Hàm chính
def main():
    # Nhập đường dẫn ảnh
    image_path = input("Nhập đường dẫn đến tệp ảnh (jpg, png, bmp): ")

    # Gọi hàm để phát hiện khuôn mặt
    detect_faces(image_path)

# Chạy chương trình
if __name__ == "__main__":
    main()
```

PHÁT TRIỂN HỆ THỐNG ĐA PHƯƠNG TIỆN

1) Trình bày mã UTF-8 1 byte cho code point 7 bits : 1101001. Miền biểu diễn UTF-8 1 byte.

==>011010001

2) Trình bày mã UTF-8 2 byte cho code point 11 bits : 01011101001. Miền biểu diễn UTF-8 2 byte.

Byte1: 11001011

Byte2: 10101001 Lấy 11 bit từ codepoint ==>1100101110101001= CBA9

3) Trình bày mã UTF-8 3 byte cho code point 16 bits :

1011 010111 110101. Miền biểu diễn UTF-8 3 byte.

Byte1: 11101011

Byte2: 10010111

Byte3: 10110101 Lấy 16 bit từ code point ==>1110101110010111110110101=EB97B5

4) Trình bày mã UTF-8 4 byte cho code point 21 bits :

110 100111 010011 101001. Miền biểu diễn UTF-8 4 byte.

Byte1: 11110110 Byte2: 10100111 Byte3: 10010011

Lấy 24 bits từ code point ==>111101101010011110010011101010001=F6A793A9

5) Trình bày 3giai đoạn Sampling, Quantization và Coding để chuyển đổi tín hiệu Analog sang Digital. Thế nào là kỹ thuật PCM Coding. Hãy cho ví dụ.

Để chuyển đổi tín hiệu Analog sang Digital, người ta thực hiện qua 3 giai đoạn: Sampling, Quantization và Coding

- Sampling (Lấy mẫu): Tín hiệu analog thực tế (như nhiệt độ, độ ẩm, âm thanh,…) được lấy mẫu theo các khoảng thời gian nhất định. Mỗi khoảng thời gian, tín hiệu âm thanh sẽ được đo và lưu trữ giá trị của nó
- Quantization (Lượng tử hóa): Các giá trị mẫu âm thanh được chuyển đổi thành các trạng thái 0 và 1.
- Ví dụ, tín hiệu có giá trị mẫu thấp hơn ngưỡng được mã hóa thành 0, trong khi tín hiệu có giá trị mẫu cao hơn ngưỡng được mã hóa thành 1.
- Coding (Mã hóa): Tín hiệu số được biểu diễn bằng các bit. Quá trình này cho phép tín hiệu âm thanh được lưu trữ, truyền và tái tạo lại một cách chính xác Kỹ thuật PCM (Pulse-Code Modulation) là một phương pháp biểu diễn kỹ thuật số của tín hiệu analog mà sẽ đưa các mẫu biên độ của tín hiệu analog đều đặn. Dữ liệu tương tự được lấy mẫu được thay đổi thành, và sau đó được biểu thị bằng dữ liệu nhị phân. PCM yêu cầu một chiếc đồng hồ rất chính xác.

Ví dụ về kỹ thuật PCM Coding: Trong công nghệ âm thanh, PCM được sử dụng rộng rãi. Ví dụ, khi bạn ghi âm bằng phần mềm ghi âm trên máy tính hoặc điện thoại, tín hiệu âm thanh từ micro sẽ được chuyển đổi từ dạng analog sang dạng số bằng kỹ thuật PCM. Sau khi ghi xong, bạn có thể lưu lại file âm thanh dưới dạng số này để nghe lại sau

6) Phát biểu định lý NYQUIST. Viết công thức và giải thích các đại lượng.

Định lý lấy mẫu Nyquist–Shannon được sử dụng trong lĩnh vực lý thuyết thông tin, đặc biệt là trong viễn thông và xử lý tín hiệu. Định lý này được phát biểu như sau:

Một hàm số tín hiệu x(t) không chứa bất kỳ thành phần tần số nào lớn hơn hoặc bằng một giá trị fm có thể biểu diễn chính xác bằng tập các giá trị của nó với chu kỳ lấy mẫu T = 1/(2fm).

Trong đó:

- x(t) là tín hiệu cần lấy mẫu.
- fm là thành phần tần số lớn nhất có trong tín hiệu. T là chu kỳ lấy mẫu.

Theo định lý, tần số lấy mẫu fs phải thoả mãn điều kiện fs ≥ 2fm1. Tần số giới hạn fs/2 này được gọi là tần số Nyquist và khoảng (-fs/2; fs/2) gọi là khoảng Nyquist. Thực tế, tín hiệu trước khi lấy mẫu sẽ bị giới hạn bằng một bộ lọc để tần số tín hiệu nằm trong khoảng Nyquist.

7) Áp dụng: Giả sử một tín hiệu Audio có dải tần số từ 1Khz đến 25 Khz. a)Xác định tần số lấy mẫu b)Xác định tốc độ Bit-Rate, biết mỗi mẫu được mã hóa bởi chuỗi 8bits, 16 bits và 32 bits.

Để xác định tần số lấy mẫu, bạn cần sử dụng công thức Nyquist. Tần số lấy mẫu ít nhất phải là gấp đôi tần số tín hiệu cao nhất. Vì vậy, tần số lấy mẫu = 2 x 25 KHz = 50 KHz. Để xác định tốc độ Bit-Rate, bạn cần biết tần số lấy mẫu. Dùng công thức Bit-Rate = Tần số lấy mẫu x Số bit trên mẫu.

- Với chuỗi 8 bits: 50 KHz x 8 bits = 400 Kbps

- Với chuỗi 16 bits: 50 KHz x 16 bits = 800 Kbps

- Với chuỗi 32 bits: 50 KHz x 32 bits = 1600 Kbps

8) Giả sử tần số tín hiệu telephone là 5000 Hz. Hãy tính tốc độ Bit Rate. Biết số bit mã hóa cho 1 mẫu telephone là 8 bits.

Theo định lý Nyquist, tần số lấy mẫu fs phải thoả mãn điều kiện fs ≥ 2fm. Trong trường hợp này, tần số cao nhất của tín hiệu là 5000 Hz, vì vậy tần số lấy mẫu tối thiểu phải là 2*5000 Hz = 10000 Hz (hoặc 10 kHz).

Tốc độ Bit-Rate được tính bằng cách nhân tần số lấy mẫu với số bit mã hóa cho mỗi mẫu. Vì vậy, trong trường hợp này, Bit-Rate = 10 kHz * 8 bits = 80000 bits/s (hoặc 80 kbps).

9) Giả sử tần số tín hiệu telephone là 5000 Hz. Hãy tính tốc độ BitRate. Biết số bit mã hóa cho 1 mẫu telephone là 8 bits.

Giống Câu 8

10) Giả sử tần số tín hiệu Radio AM là 6515 Hz. Hãy tính tốc độ Bit Rate. Biết số bit mã hóa cho 1 mẫu Radio AM là 8 bits.

Theo định lý Nyquist, tần số lấy mẫu fs phải thoả mãn điều kiện fs ≥ 2fm. Trong trường hợp này, tần số cao nhất của tín hiệu là 6515 Hz, vì vậy tần số lấy mẫu tối thiểu phải là 2*6515 Hz = 13030 Hz (hoặc 13.03 kHz).

Tốc độ Bit-Rate được tính bằng cách nhân tần số lấy mẫu với số bit mã hóa cho mỗi mẫu. Vì vậy, trong trường hợp này, Bit-Rate = 13.03 kHz * 8 bits = 104240 bits/s (hoặc 104.24 kbps).

11) Giả sử tần số tín hiệu Radio FM là 11.05 KHz. Hãy tính tốc độ Bit Rate. Biết số bit mã hóa cho 1 mẫu Radio FM là 32 bits.

Theo định lý Nyquist, tần số lấy mẫu fs phải thoả mãn điều kiện fs ≥ 2fm. Trong trường hợp này, tần số cao nhất của tín hiệu là 11.05 kHz, vì vậy tần số lấy mẫu tối thiểu phải là 2*11.05 kHz = 22.1 kHz.

Tốc độ Bit-Rate được tính bằng cách nhân tần số lấy mẫu với số bit mã hóa cho mỗi mẫu. Vì vậy, trong trường hợp này, Bit-Rate = 22.1 kHz * 32 bits = 707200 bits/s (hoặc 707.2 kbps).

12) Giả sử tần số tín hiệu CD là 22.05 KHz. Hãy tính tốc độ Bit-Rate. Biết số bit mã hóa cho 1 mẫu CD là 32 bits.

Theo định lý Nyquist, tần số lấy mẫu fs phải thoả mãn điều kiện fs ≥ 2fm. Trong trường hợp này, tần số cao nhất của tín hiệu là 22.05 kHz, vì vậy tần số lấy mẫu tối thiểu phải là 2*22.05 kHz = 44.1 kHz.

Tốc độ Bit-Rate được tính bằng cách nhân tần số lấy mẫu với số bit mã hóa cho mỗi mẫu. Vì vậy, trong trường hợp này, Bit-Rate = 44.1 kHz * 32 bits = 1411200 bits/s (hoặc 1411.2 kbps).

13) Giả sử tần số tín hiệu dữ liệu là 5 KHz. Người ta thực hiện lấy mẫu dữ liệu này và nhận thấy số mức lượng hóa tối đa là 249 mức.

a)Hỏi số bit tối thiểu dùng để mã hóa số mức tối đa này là bao nhiêu?

b)Hãy tính tốc độ Bit-Rate.

a) Số bit tối thiểu dùng để mã hóa số mức tối đa này: Để biểu diễn 249 mức khác nhau, chúng ta cần sử dụng log2(249) bit. Khi tính toán, chúng ta nhận được kết quả là khoảng 7.96. Tuy nhiên, số bit phải là một số nguyên, vì vậy chúng ta cần làm tròn lên và kết quả là 8 bit.

b) Tốc độ Bit-Rate: Tốc độ Bit-Rate được tính bằng cách nhân tần số lấy mẫu với số bit mã hóa cho mỗi mẫu. Theo định lý Nyquist, tần số lấy mẫu fs phải thoả mãn điều kiện fs ≥ 2fm. Trong trường hợp này, tần số cao nhất của tín hiệu là 5 kHz, vì vậy tần số lấy mẫu tối thiểu phải là 2*5 kHz = 10 kHz. Vì vậy, Bit-Rate = 10 kHz * 8 bits = 80000 bits/s (hoặc 80 kbps).

14) Giả sử tần số tín hiệu dữ liệu là 20 KHz. Người ta thực hiện lấy mẫu dữ liệu này và nhận thấy số mức lượng hóa tối đa là 64,512 mức. a)Hỏi số bit tối thiểu dùng để mã hóa số mức tối đa này là bao nhiêu? b)Hãy tính tốc độ Bit-Rate.

a) Số bit tối thiểu cần để mã hóa số mức tối đa 64,512 mức là:

⌈log2(64512)⌉=16 bit

b) Tốc độ Bit-Rate có thể được tính như sau:

Bit-Rate=20×10^3×16=320 Mbps Vậy, tốc độ Bit-Rate là 320 Mbp

15) Giả sử tần số tín hiệu dữ liệu là 22 KHz. Người ta thực hiện lấy mẫu dữ liệu này và nhận thấy số mức lượng hóa tối đa là 4,294,976,124 mức. a)Hỏi số bit tối thiểu dùng để mã hóa số mức tối đa này là bao nhiêu? b)Hãy tính tốc độ Bit-Rate

a) Số bit tối thiểu cần để mã hóa số mức tối đa 4,294,976,124 mức là:

$$\lceil \log_2(4294976124) \rceil = 32 \text{ bit}$$

b) Tốc độ Bit-Rate có thể được tính như sau:

Bit-Rate=số bit tối thiểu x với tần số lấy mẫu= 22×10^3×32=704 Mbps

Vậy, tốc độ Bit-Rate là 704 Mbps.

Digitizing Audio : Khi âm thanh thu vào micro, các tín hiệu âm thanh được chuyển đổi thành các tín hiệu điện. Tín hiệu tương tự(analog digital) có thể chuyển đổi thành các tín hiệu số(digital signal) để phục vụ với nhiều mục đích.

Theo định lý Nyquist, fs >= 2*f

với f là tần số lớn nhất của băng tần tín hiệu audio, fs là tần số lấy mẫu…

Vd: một tín hiệu audio được lấy mẫu với tốc độ 8000 mẫu/s; mỗi mẫu

được mã hóa bởi chuỗi 8 bit. Tính tốc độ truyền dữ liệu(truyền bit). Ta có: R=B x fs => 8000 x 8 = 64000 bp

----------/\----------

LÝ THUYẾT

Màu sắc là cảm giác được ghi nhận khi não bộ cảm nhận được ánh sáng có bước sóng khác nhau.

• Người ta quan sát thấy được ở các vật thể xung quang là do vật thể phản xạ hoặc phát ra các bước sóng ánh sáng nhất định.

• Có thể tạo ra cảm giác của bất kỳ màu nào bằng cách trộn một lượng thích hợp của ba màu cơ bản: đỏ- xanh lá cây- xanh lam.

• Có thể tạo màu trên màn hình máy tính bằng cách sử dụng sự phát ra của ba bước sóng ánh sáng trong các kiểu kết hợp thích hợp.

• Hue : phân biệt giữa các màu như đỏ, xanh lá cây và vàng.

• Độ bão hòa đề cập đến khoảng cách màu sắc so với màu xám có cường độ tương đương.

• Độ sáng(lightless) thể hiện khái niệm không sắc về cường độ cảm nhận của một đối tượng phản xạ.

• Độ chói sáng(brightness) được sử dụng thay vì độ sáng cho một vật thể tự phát sáng chẳng hạn như CRT.

Điểm ảnh(pixel): thành phần cơ bản trong hình ảnh kỹ thuật số Độ phân giải hình ảnh (M * N): số pixel trong hình ảnh kỹ thuật số Mô hình màu RGB: mỗi màu hiển thị được mô tả gồm các tham số độc lập - độ chói của các tham số chính được sử dụng trong màn hình CRT màu.

Phân biệt các ảnh monochrome, Gray-scale, 8 bits color và 24 bits Color.

1. **Ảnh Monochrome:** Đây là loại ảnh chỉ sử dụng một màu duy nhất để biểu diễn. Thông thường, nó sẽ là ảnh đen trắng. Mỗi điểm ảnh trong ảnh monochrome có thể được biểu diễn bằng một bit duy nhất (0 hoặc 1), trong đó 0 thường là màu đen và 1 là màu trắng.

2. **Ảnh Gray-scale:** ảnh grayscale cũng chỉ sử dụng các mức độ của sáng độ để biểu diễn, nhưng có thể có nhiều hơn một màu sáng tối khác nhau. Ảnh grayscale thường sử dụng 8 bits (1 byte) cho mỗi điểm ảnh, cho phép 256 mức sáng khác nhau từ đen đến trắng.

3. **8-bit Color:** Ảnh này sử dụng 8 bits cho mỗi pixel để biểu diễn màu sắc. Với 8 bits, mỗi kênh màu (RGB - đỏ, xanh lá, và lam) có thể có 256 mức độ khác nhau, cho tổng cộng khoảng 16.7 triệu màu (256^3).

4. **24-bit Color:** Ở đây, mỗi pixel được biểu diễn bằng 24 bits (3 bytes). Cấu trúc này gồm 8 bits cho mỗi kênh màu (RGB), điều này cho phép 16.7 triệu màu (cũng như 8-bit color), nhưng sử dụng cách biểu diễn khác nhau.

Chroma subsampling: hệ thống thị giác của con người nhạy cảm hơn với độ sáng(luminance) so với độ chói(chrominance)

Phân biệt các mô hình subsampling: 4:4:4 (no sampling) ; 4:2:2 ; 4:1:1 ; 4:2:0,…

1. **4:4:4 (No subsampling):** Trong mô hình này, mỗi kênh màu (ví dụ: RGB - đỏ, xanh lá, lam) đều được lưu trữ một cách đầy đủ cho mỗi pixel. Điều này có nghĩa là mọi pixel đều có thông tin về tất cả các kênh màu, không có việc giảm bớt thông tin màu sắc.

2. **4:2:2:** Ở đây, mẫu màu sắc được giảm xuống còn 50% theo chiều ngang (horizontal) so với 4:4:4. Cụ thể, thông tin về màu sắc chỉ được lưu trữ cho mỗi cặp pixel theo chiều ngang, trong khi các pixel cùng hàng dọc vẫn chia sẻ cùng thông tin về màu sắc.

3. **4:1:1:** Trong mô hình này, mẫu màu sắc bị giảm xuống 75% theo chiều ngang và chỉ 25% theo chiều dọc so với 4:4:4. Thông tin về màu sắc chỉ được lưu trữ cho mỗi bốn pixel.

4. **4:2:0:** Đây là mô hình subsampling phổ biến trong video và ảnh số. Màu sắc được giảm xuống 50% theo cả chiều ngang và chiều dọc so với 4:4:4. Trong khi thông tin về màu sắc của mỗi hàng vẫn được lưu trữ, thông tin về màu sắc của mỗi hai hàng liên tiếp chỉ được lưu trữ cho một hàng.

Công thức tính tốc độ truyền video:

Bit Rate = width * height * depth * fps (bits/sec) **bps**

Digitizing Video: Video là chuỗi các frame hiển thị liên tục. Nếu các frame hiển thị đầy đủ với tốc độ phù hợp, thì người xem sẽ cảm nhận ấn tượng với các hình ảnh chuyển động. Tốc độ frame thay đổi tùy theo chuẩn : vd: Bắc Mỹ 25 frames/s… Mỗi frame được chia thành các lưới nhỏ(block), mỗi phần tử gọi là pixel. Đối với TV trắng-đen, mỗi pixel 8 bit trình bày một trong 256 mức xám. Đối với TV màu, mỗi pixel 24 bit trình bày 8 bit cho mỗi màu red, green và blue…

Video Compression : MPEG là chuẩn nén video, trong đó hình ảnh chuyển động là chuỗi các frame hiển thị lần lượt. Mỗi frame là tập hợp nhiều pixel, còn video là chuỗi các frame tổ hợp theo thời gian…

Spatial Compression: mỗi frame được thực hiện với chuẩn nén JPEG, mỗi frame được nén độc lập.

Temporal Compression: các frame dư thừa sẽ bị loại. Khi xem TV, chúng ta nhận được 50 frames/s. Tuy nhiên, hầu hết các frame liên tục thì tương tự. Ví dụ: Khi đang nói, hầu hết các frame thì tương tự như các frame trước, ngoại trừ phân đoạn của frame xung quanh miệng.

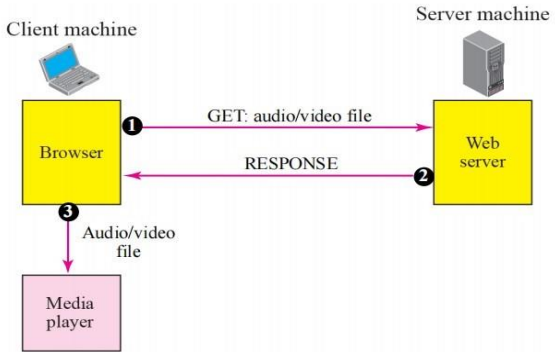
Phương pháp MPEG chia các frame vào 3 loại: I-frames, P-frames và B-frames I-frames: (intracoded frame) là frame độc lập, không liên quan đến bất kỳ frame nào khác. I-frames không được cấu trúc lại từ các frame khác… P-frames: (predicted frame) : liên hệ đến I-frame cho trước hay P frame. Mỗi P-frame chỉ cấu trúc lại từ các frame I và P đã có trước B-frames: (bidirectional frame) liên hệ đến frame I và P trước và sau. Nhưng B-frames không liên quan đến B-frames khác.

MPEG có 2 phiên bản: CD-ROM với tốc độ 1.5 Mbps và High quality DVD với tốc độ 6Mbps

LO2 : Giải thích được các công nghệ (hệ điều hành, mạng) hỗ trợ hệ thống đa phương tiện

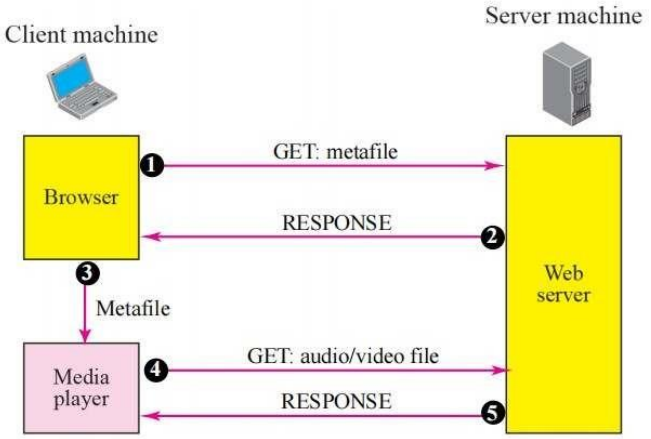
Vẽ hình và trình bày 4 mô hình STREAMING STORED AUDIO/VIDEO:

WEB SERVER



Tệp âm thanh/video nén có thể được tải xuống dưới dạng tệp văn bản. Máy khách (trình duyệt) có thể sử dụng dịch vụ HTTP và gửi tin nhắn GET để tải tập tin xuống. Máy chủ Web có thể gửi nén vào trình duyệt. Sau đó, trình duyệt có thể sử dụng một ứng dụng trợ giúp, thường được gọi là media player, để phát tệp. Tệp cần được tải xuống hoàn toàn trước khi có thể phát.

WEB SERVER WITH METAFILE



1. Máy khách HTTP truy cập máy chủ Web bằng thông báo GET.

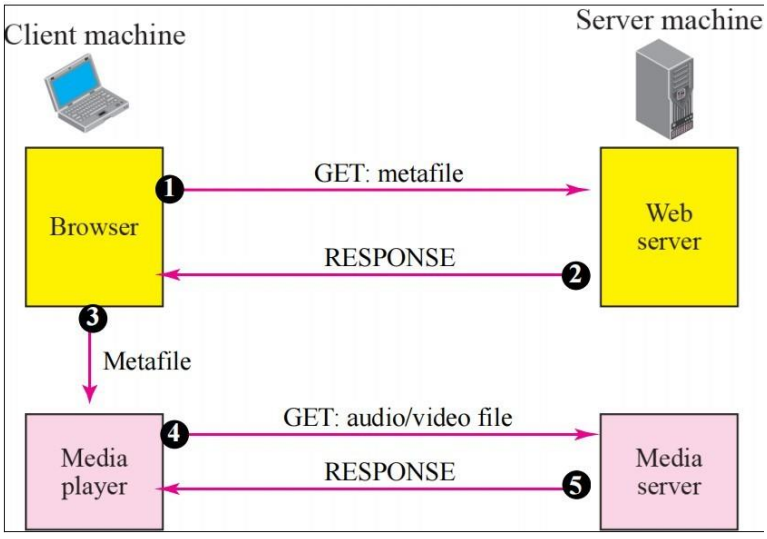
2. Thông tin về siêu tệp sẽ có trong phản hồi.

3. Siêu tệp được chuyển đến trình phát đa phương tiện.

4. Trình phát đa phương tiện sử dụng URL trong siêu tệp để truy cập tệp âm thanh/video.

5. Máy chủ Web phản hồi.

MEDIA SERVER



1. Máy khách HTTP truy cập máy chủ Web bằng thông báo GET.

2. Thông tin về siêu tệp sẽ có trong phản hồi.

3. Siêu tệp được chuyển đến trình phát đa phương tiện.

4. Trình phát đa phương tiện sử dụng URL trong siêu tệp để truy cập vào máy chủ phương tiện để tải tệp xuống.

Việc tải xuống có thể diễn ra bằng bất kỳ giao thức nào sử dụng UDP.

5. Máy chủ phương tiện phản hồi.

MEDIA SERVER and RTSP

1. Máy khách HTTP truy cập máy chủ Web bằng thông báo GET.

2. Thông tin về siêu tệp sẽ có trong phản hồi.

3. Siêu tệp được chuyển đến trình phát đa phương tiện.

4. Trình phát media gửi thông báo CÀI ĐẶT để tạo kết nối với máy chủ media.

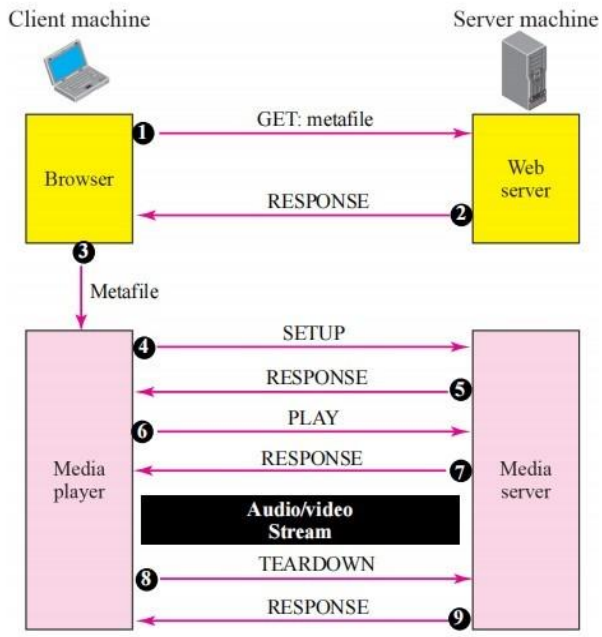
5. Máy chủ phương tiện phản hồi.

6. Trình phát media gửi tin nhắn PLAY để bắt đầu phát (tải xuống).

7. Tệp âm thanh/video được tải xuống bằng giao thức khác chạy trên UDP.

8. Kết nối bị ngắt khi có thông báo TEARDOWN.

9. Máy chủ phương tiện phản hồi.



RTSP: Real-Time Streaming Protocol là giao thức thực hiện một số chức năng xử lý chuỗi. RTSP điều khiển quá trình hiển thị nội dung Audio/video…

REAL-TIME INTERACTIVE AUDIO/VIDEO

Time Ralationship: Giả sử server tạo ra dữ liệu video và gởi đi. Video được số hóa và đóng gói. Mỗi gói có kích thước hiển thị là 10s, gói đầu tiên bắt đầu ở thời điểm là 00:00:00, gói 2 là 00:00:10 và gói 3 là 00:00:20. Giả sử thời gian truyền mỗi gói mất 1s để đến đích, nghĩa là nơi nhận có thể hiển thị nội dung ở thời điểm 00:00:01 (gói 1),…

Hiện tượng jitter là gì?

Phương sai độ trễ là sự thay đổi thời gian trễ giữa hoạt động truyền dẫn dữ liệu và nhận dữ liệu qua một kết nối mạng. Để người dùng có trải nghiệm tốt hơn, độ trễ nhất quán sẽ được ưu tiên hơn so với độ trễ có tính biến động.

Trình bày và so sánh 2 giao thức SIP & H.323 Giao thức SIP (Session Initiation Protocol) :

Định nghĩa: là một giao thức báo hiệu được sử dụng để thiết lập một phiên giao dịch trực tuyến giữa 2 hoặc nhiều người tham gia, sửa đổi phiên đó và cuối cùng chấm dứt phiên đó.

các SIP messges(INVITE,ACK,BYE,OPTIONS,CANCEL,REGISTER)

1. **INVITE:** Mời một người dùng hoặc thiết bị tham gia vào một cuộc gọi.
2. **ACK (Acknowledgment):** Xác nhận rằng một thông điệp đã được nhận.
3. **BYE:** Kết thúc cuộc gọi.
4. **OPTIONS:** Kiểm tra khả năng của một thiết bị hoặc máy chủ.
5. **CANCEL:** Hủy bỏ cuộc gọi đang được thực hiện.
6. **REGISTER:** Đăng ký vị trí của một thiết bị với máy chủ SIP.

vẽ hình SIP Session, trình bày chi tiết 3 giai đoạn Establishing a Session, Communicating, và Terminating the Session.

Caller | Callee |

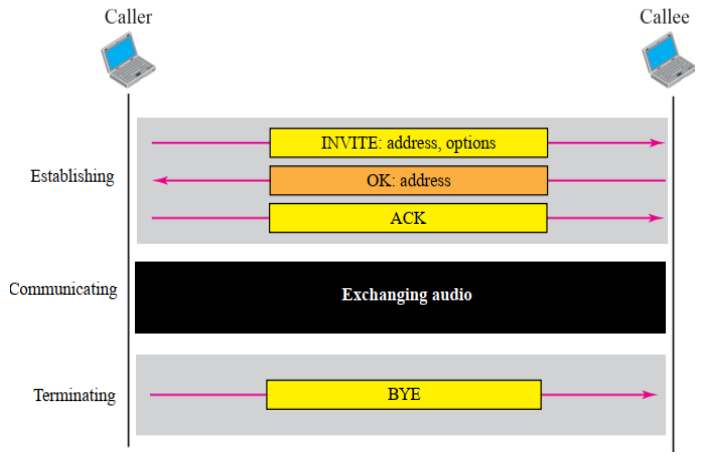
INVITE (Mời) ----------> |

| INVITE (Mời)

| <---------- |

| 200 OK (Đồng ý) |

ACK (Xác nhận) ----------> |



1 Establishing a Session

- **Người gọi (Caller)** gửi INVITE đến người được gọi (Callee) để mời họ tham gia cuộc gọi.
- **Callee** trả lời INVITE với 200 OK để xác nhận.
- Người gọi gửi ACK để xác nhận đồng ý của Callee và bắt đầu cuộc gọi.

2. Communicating

Sau khi phiên đã được thiết lập, việc truyền thông diễn ra giữa Caller và Callee. Truyền thông có thể bao gồm truyền dữ liệu âm thanh, video hoặc dữ liệu đa phương琯椀ện khác.

3. Terminating the Session.

- Caller gửi BYE để kết thúc cuộc gọi.
- Callee trả lời bằng 200 OK để xác nhận và kết thúc cuộc gọi.

Giao thức H.323:

¬ Cho phép mạng điện thoại công cộng giao tiếp với hệ thống máy

tính kết nối Internet.

¬ Vẽ hình mô tả kiến trúc H.323

¬ Mô tả giao thức H.323 ¬ Vẽ hình và mô tả hoạt động H.323.

LO3: Phát triển ứng dụng đa phương tiện trên nền tảng web hoặc peer to-peer

Xem le pdf: MULTIMEDIA_AND_WEB_TEACH.pdf Đa phương tiện

- Sự tích hợp của nhiều loại phương tiện, chẳng hạn như văn bản, hình ảnh, video, hoạt ảnh và âm thanh

- Đa phương tiện dựa trên Web (gọi tắt là đa phương tiện)

- Đa phương tiện (âm thanh, video, hình ảnh động) nằm trên các trang Web

• Các trang web đa phương tiện

- Có tính tương tác

- Thường chứa các phần tử mà người dùng tương tác trực tiếp

- Hiển thị thông tin theo yêu cầu của khách truy cập trang Web

Đa phương tiện dựa trên Web là gì?

• Công nghệ máy tính phát triển và kết nối Internet băng thông rộng làm cho Đa phương tiện dựa trên web khả thi hơn nhiều so với trước đây

• Phần lớn các trang Web ngày nay bao gồm đa phương tiện như quảng cáo, chương trình truyền hình, podcast, nội dung do người dùng tạo ra

• Tìm hiểu về Đa phương tiện dựa trên Web?

- Đa phương tiện là một thành phần không thể thiếu của Web- Các doanh nghiệp và cá nhân cần hiểu đặc điểm của các loại phần tử đa phương tiện và tác động của việc thêm chúng vào trang Web

Web-Based Multimedia Applications (chú ý cho một số ví dụ mà A/C biết, với tên gọi cụ thể như Youtube,…)

•Cung cấp thông tin

- Hình ảnh về sản phẩm, video clip và podcast cũng như của người dùng hướng dẫn sử dụng được sử dụng để truyền đạt thông tin.

- Thành phần quan trọng trong việc đào tạo, huấn luyện, dạy học,… dựa trên Web (WBT) .

•Thương mại điện tử

- Danh mục trực tuyến, mẫu phim và nhạc, v.v.

- Thực tế ảo (VR)

• Việc sử dụng máy tính để tạo ra không gian ba chiều môi trường trông giống như trong thế giới thực(vd: thiết kế kiến trúc, dạy học mô phỏng,…)

• Thực tế ảo tăng cường – hỗ trợ hình ảnh theo thời gian thực

•Giải trí

- Truyền hình trực tuyến / phim và trò chơi có sẵn qua TV các trang mạng

• Truyền thông xã hội và thế giới ảo

- Hình ảnh và video trên nhiều trang mạng xã hội- Thế giới ảo 3D(Đời sống thứ hai)