

# CS 100: Compression Wrap-up, The Internet

Chris Kauffman

Week 9-2

# Logistics

## HW 5

- ▶ Due next week Friday
- ▶ Simple Encryption Problems
- ▶ Creating your own personal web page

## Today

- ▶ Wrap up compression discussion
- ▶ How does the internet work?

## Reading

- ▶ "Pattern": Chapter 6
- ▶ "Zyante": Ch 5

## Compressing Images: Lossy

10x Compression: 39kb



50x Compression: 9kb



# Sound and Video Compression

- ▶ Sound is a continuous phenomenon and is always discretized for digital storage
  - ▶ Not so for records and magnetic tapes
- ▶ WAV or .wav files are typically raw audio
  - ▶ Files are typically very large,
  - ▶ 70-minute CD would take about 600 MB of space
- ▶ FLAC is a **lossless** audio compression format
  - ▶ Can cut down size of files by 20-30%
  - ▶ Quality of sound does not suffer
- ▶ MP3 and AAC are **lossy** compression formats
  - ▶ Can fit a whole CD into 30-40 MB
  - ▶ Quality of sound does change
  - ▶ Can choose the level of quality when compressing
  - ▶ Higher quality requires more space
- ▶ MP4 is a **lossy video** format
  - ▶ Combines techniques from MP3 and JPEG
  - ▶ From one picture to the next, records changes in pixels
  - ▶ Often compresses raw video/audio by 100x with little perceptible loss in quality

## Information Density

176K The-Time-Machine.txt  
70K The-Time-Machine.zip

- ▶ Which one has more **information**?
- ▶ *What is information?*

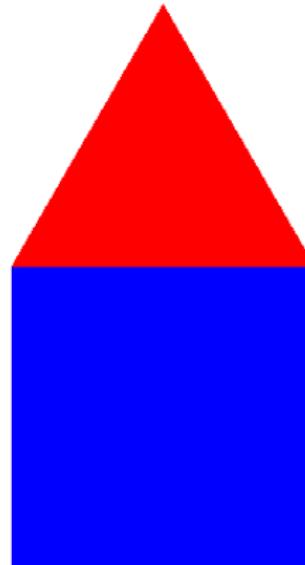
# Small Programs that Generate Pictures

Which is smaller?

## Python Code

```
from turtle import *
def house(size):
    color("blue")
    begin_fill()
    for i in range(4):
        forward(size)
        right(90)
    end_fill()
    color("red")
    begin_fill()
    right(300)
    for i in range(3):
        forward(size)
        right(120)
    end_fill()
# Draw a big house
house(200)
hideturtle()
```

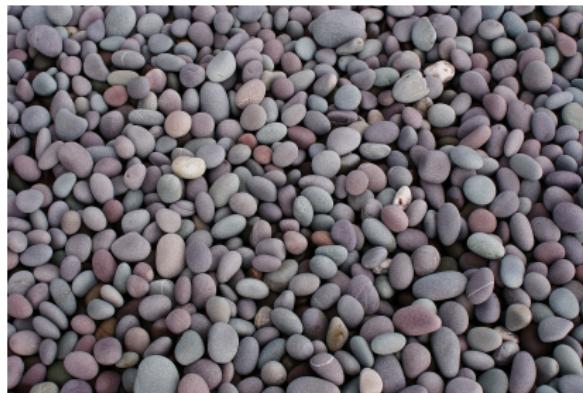
## Picture



# An Interesting Measure of Information

Information Content is equal to  
the smallest computer program to  
re-create the data

- ▶ For *The Time Machine*,  
would look something like  
Huffman Encoding of text
- ▶ *Pattern on the Stone*: For  
pebbles on the beach, things  
get interesting
- ▶ How exact must the  
recreation be?

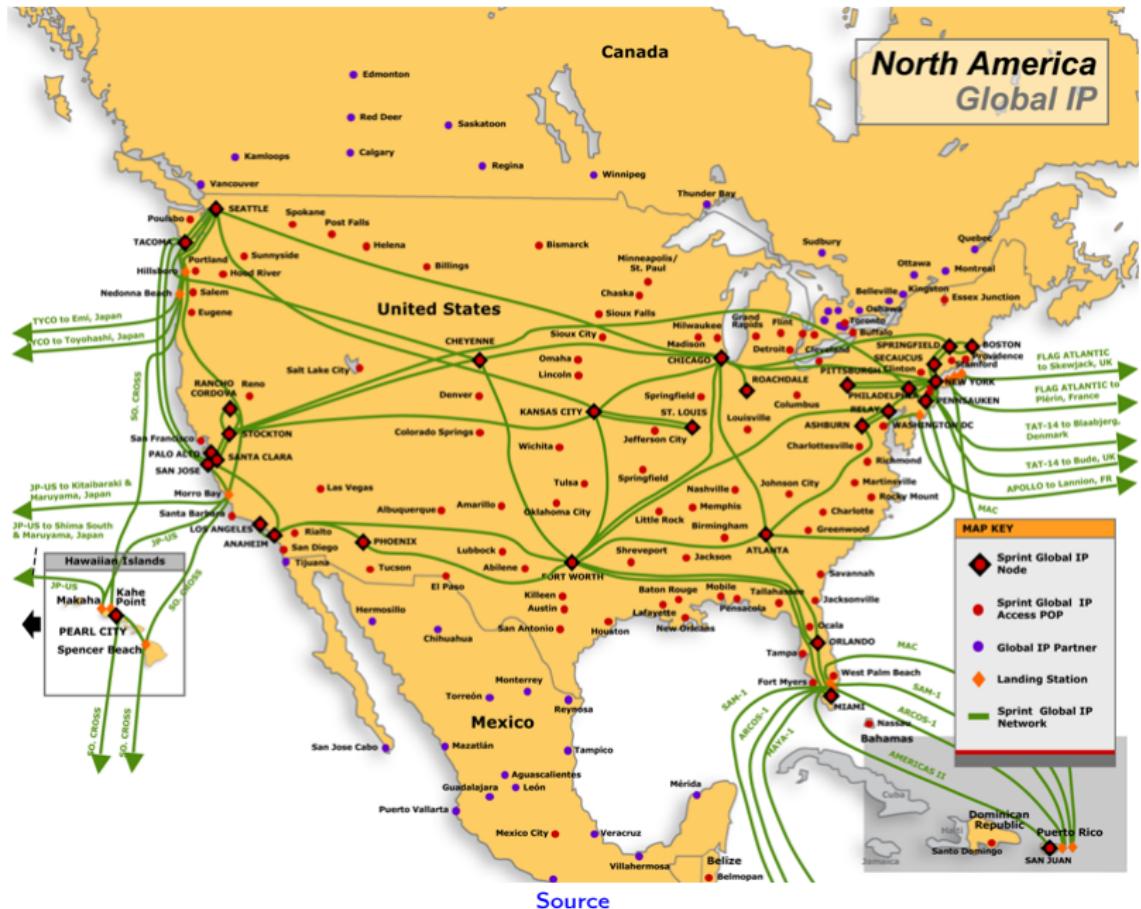


Source

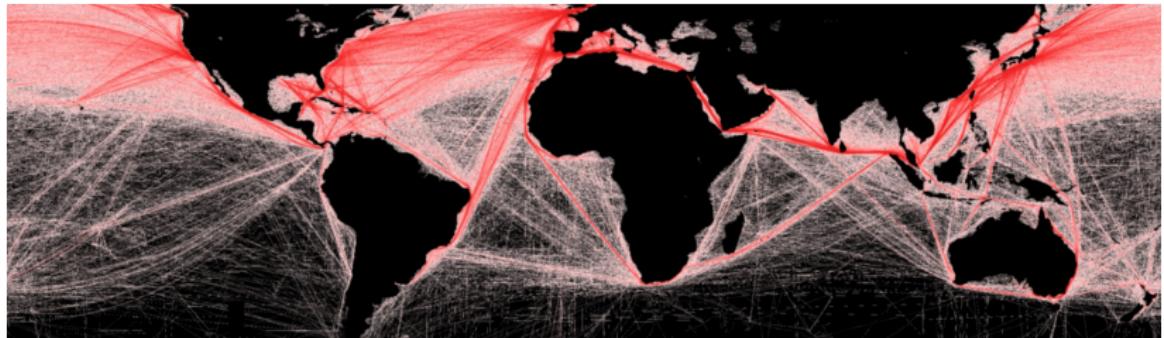
## How does the Internet work?

- ▶ A lot of wires
- ▶ An older analogous system
- ▶ An interesting set of problems

## Backbone Example: Sprint



# The Backbone Under the Sea

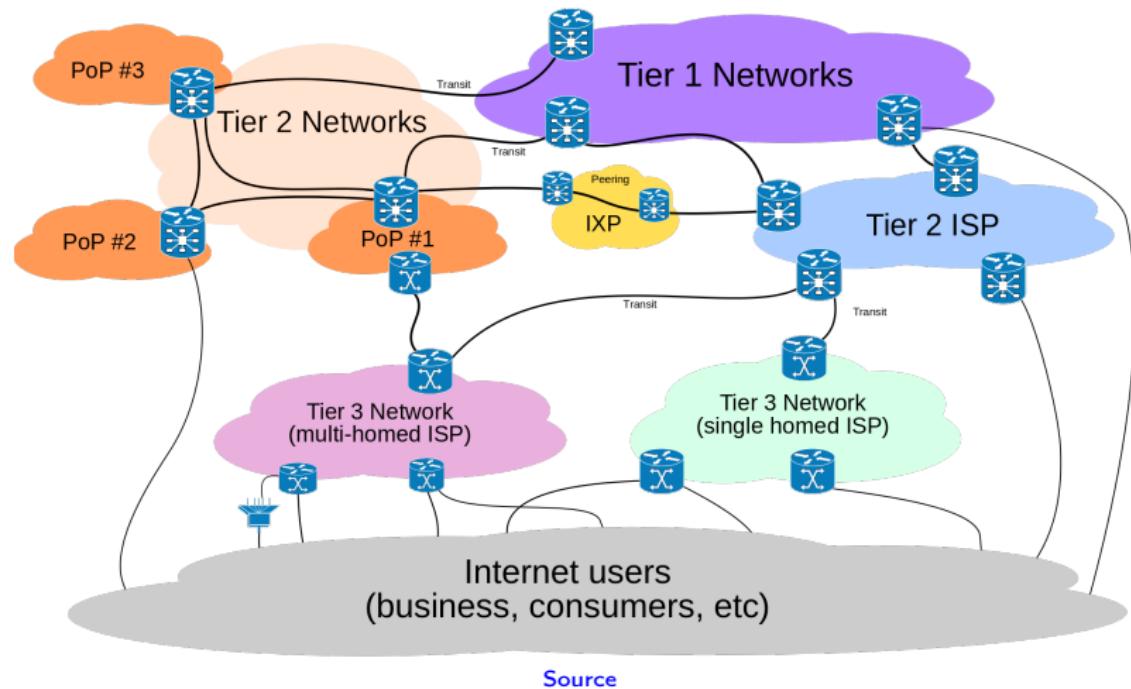


Attributed to B.S.Halpern (T.Hengl; D.Groll)



Source

# Layers of Networks



# Python Can Read Web Stuff

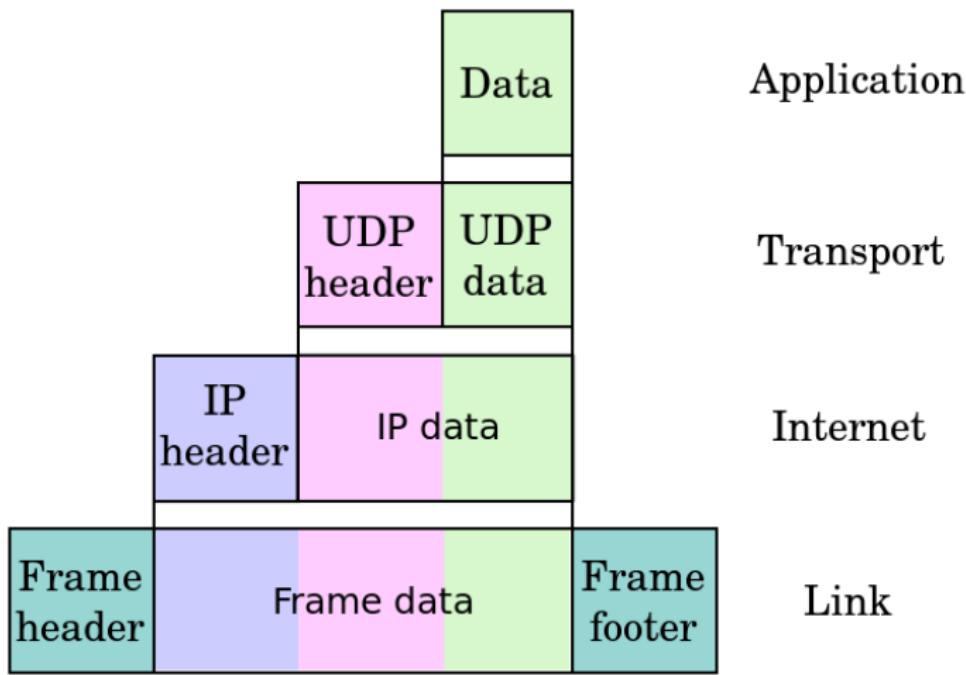
The `urllib.request` package is very useful for this

```
from urllib.request import *

url = "http://www.google.com"      # Where to connect
connection = urlopen(url)         # connect to google
bytes = connection.read()         # read whole page
text = bytes.decode("UTF-8")        # decode to text
print(text)                       # print

url = "http://www.cs.gmu.edu/~kauffman/jazz-albums.txt"
# Merge several steps
jazz = urlopen(url).read().decode("UTF-8")
print(jazz)
```

# Layers of Communication



Source

# Putting Stuff on the Internet

You can put stuff on the internet

- ▶ Not on Facebook, Twitter, Tumblr, Flickr, etc
- ▶ On your own **personal web page**
- ▶ For HW 5, will need to do just that
- ▶ [Instructions here](#)

## General process to put stuff on the web

1. Have an account on a server: `mason.gmu.edu` (all of you do)
2. Know your username and password on server: NetID and pass for GMU
3. Log into the server using a secure FTP connection
  - ▶ Mac OS X: Terminal + sftp command
  - ▶ Windows: [Download Putty](#) and install, use PSFT ([putty home](#))
4. Issue commands to set up a web directory on the server
5. Create files on your local computer
6. Transfer files from your local computer to the web folder on the server

More detailed explanations are on HW 5 specification

## Files to Put on The Web

Text files are great, such as this one:

<http://mason.gmu.edu/~ckauffm2/jazz-albums.txt>

But most of us prefer HTML:

<http://mason.gmu.edu/~ckauffm2/index.html>