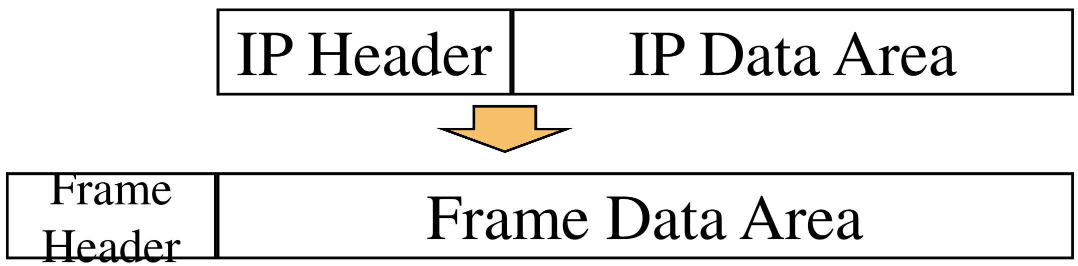
# Announcements

- Reading: Chapter 16
- Project #6 Due on Tuesday 5:00 pm
- Final is next Friday at 8:00 am (this room)

# Encapsulation

How do we send higher layer packets over lower layers?

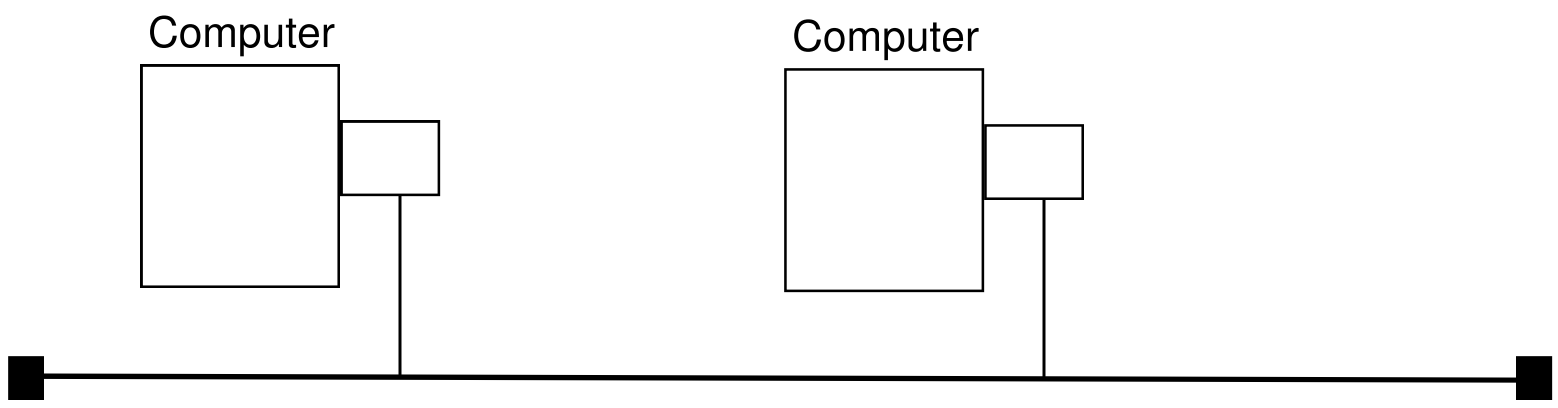
- Higher level info is opaque to lower layers
  - it's just data to be moved from one point to another




- Higher levels may support larger sizes than lower
  - could need to *fragment* a higher level packet
    - split into several lower level packets
    - need to re-assemble at the end
  - examples:
    - ATM cells are 48 bytes, but IP packets can be 64K
    - IP packets are 64K, but files are megabytes

# Ethernet

- 10 Mbps (to 100 Gbps)
- mili-second latency
- limited to several kilometers in distance
- variable sized units of transmission
- Conceptually a bus based protocol
  – requests to use the network can collide
- addresses are 48 bits
  – unique to each interface



# Switched Ethernet

- Logically it is still a bus
- Physically, it is a star configuration
  – the hub is at the center of the network
- Switches provide:
  – better control of hosts
    - possible to restrict traffic to only the desired target
    - can shutdown a host's connection at the hub if its Ethernet device is misbehaving
  – easier wiring
    - can use twisted pair wiring
- 100 Mbps/1Gbps Ethernet
  – is only available with switches
- 10Gbps Ethernet
  – Requires cat-6 (to 100 feet) or cat-7 wiring (to 100 meters)

# Ethernet Collisions

- If one host is sending, other hosts must wait
  - called Carrier Sense with Multiple Access (CSMA)
- Possible for two hosts to try to send at once
  - each host can detect this event (cd- Collision Detection)
  - both hosts must re-send information
    - if they both try immediately, will collide again
    - instead each waits a random interval then tries again
- Only provides statistical guarantee of transmission
  - however, the probability of success if higher than the probability of hardware failures and other events

# My Research Interests

- Parallel Computing
  - There are limits to how fast one processor can run
  - solution: use more than one processor
- Issues in parallel computing design
  - do the processors share memory?
    - is the memory “uniform”?
    - how do processors cache memory?
  - if not how do they communicate?
    - message passing
    - what is the latency of message passing

# Parallel Processing

- What happens in parallel?
- Several different processing steps
  - pipeline
  - simple example: grep foo | sort > out
  - called: *multiple instruction multiple data* (MIMD)
- The same operation
  - every processor runs the same instruction (or no-instruction)
  - called: *single instruction multiple data* (SIMD)
  - good for image processing
- The same program
  - every processor runs the same program, but not “lock step”
  - called: *single program multiple data* (SPMD)
  - most common model

# Issues in effective Parallel Computation

- Getting enough parallelism
  - Limited by what is left serial
  - Even 10% serial limited to a speedup of 10x even with infinite numbers of processors
- Load balancing
  - every processor should to have some work to do.
- Latency hiding/avoidance
  - getting data from other processors (or other disks) is slow
  - need to either:
    - hide the latency
      - processes can "pre-fetch" data before they need it
      - block and do something else while waiting
    - avoid the latency
      - use local memory (or cache)
      - use local disk (of file buffer cache)
- Limit communication bandwidth
  - use local data
  - use "near" data (i.e. neighbors)

# My Research:

- Given a parallel program and a machine
- Try to answer performance related questions
  - Why is the programming running so slowly?
  - How do I fix it?
- Issues:
  - how to measure a program without changing it?
  - how do you find (and then present) the performance problem, not tons of statistics?
- Techniques:
  - dynamic data collection
  - automated search
  - analysis of process interactions

# Introduction

- Software today
  - makes extensive use of libraries and re-usable components
  - Libraries used by an application may not be tuned to the application's need
- Fast software development/distribution with built-in (default) configurations
  - Applications may not run well in all environments
  - There may be no single configuration good for all environments

# Large Scale Computing

- Today (11/2014)
  - 29  systems with more than 128k processors
  - More than 50 systems >= 16k processors
  - World's fastest computer (Tianhe-2i n China)
    - 3,120,000 cores
    - Uses 17.8 MW of electricity