# Applications

- Goal: Look at two network applications using unicast and multicast
- Concepts introduced: unicast, broadcast, multicast, anycast
- Sockets, network addresses, connection oriented and connection less, TCP/UDP

- You should form your project groups (of two) now

# Educational Experimental Systems Lab

- Cushing 208. Use 4125 to enter the room
- We can use the machines in the back:
  - Gateway13-gateway18.cse.nd.edu
    - 2.6 GHz x86, 512 MB
  - Expsys-svr2.cse.nd.edu ($40K)
    - 2 processor Itanium2 server
    - 8 GB memory
  - Expsys-svr4.cse.nd.edu ($80K)
    - 4 processor Itanium2 server
    - 8 GB memory
- You can experiment and try new network technologies. They run RedHat Linux. No illegal work
- Planetlab access - more later

# www.planet-lab.org

- 512 nodes over 240 sites
- Runs Linux


# Network fundamentals

- Each machines has a name. Networks use a name (Internet address). We use DNS (Domain Name System) to convert from user friendly names to IP address
  - www.nd.edu is really 129.74.250.90
- Machines can be little-endian or big-endian. Convert all shorts and longs to network order using htons() and htonl() (or ntohs() and ntohl())

# Network fundamentals

- Connection oriented vs Connectionless
  - Connection oriented, e.g. telephone
    - First establish connection. Wait till connection establishment is completed
    - Once connection setup is completed, no need to send address/dialup
  - Connection-less, e.g. postal letter
    - No need to wait for connection setup
    - Each message should carry the address
- Reliable (TCP) vs Unreliable (UDP)

# Communication mechanisms

- Unicast: Communication between a source and a destination
- Broadcast: One sender, all receivers
  - Doesn't scale. Imagine an ability to broadcast to everyone at ND. If everyone uses this, then there is cacophony
- Multicast: One sender, many receivers
  - Only interested listeners would be bothered
  - Sender doesn't know who is listening - could be all, few or none
- Anycast: One sender, one of many receivers
  - www.google.com does have to be a single machine

# Client-Server

- Server waits in SERVER_PORT
- Client connects to Server and sends a message. Server will immediately respond back with the message

- client.c
- server.c
- utils.h
- Makefile

# Multicast

- Client will send to multicast address
- Anyone who is waiting for the multicast group can see this message
- mclient.c
- multicast.c