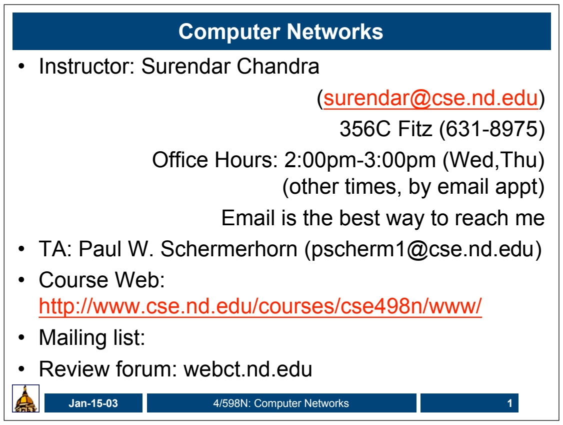
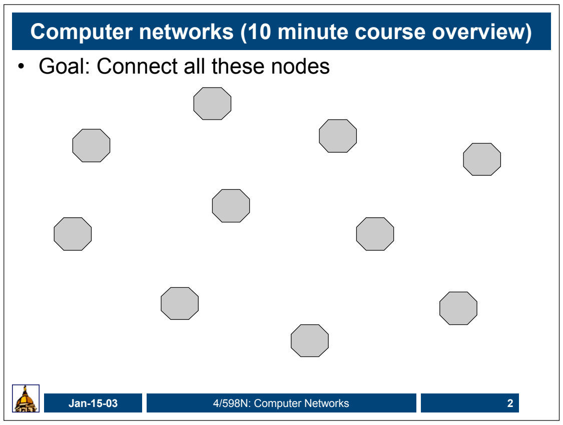


Computer Networks

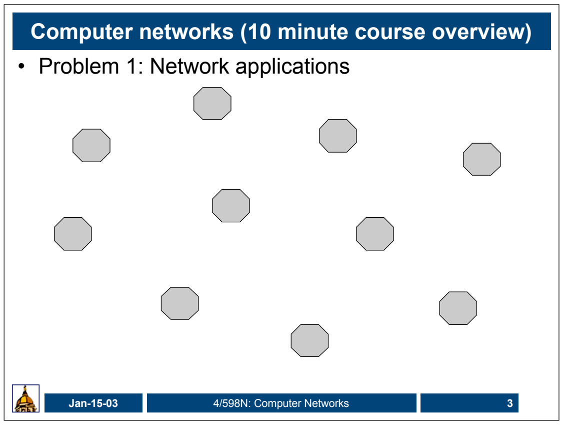
- Instructor: Surendar Chandra

  (surendar@cse.nd.edu)
  356C Fitz (631-8975)
  Office Hours: 2:00pm-3:00pm (Wed,Thu)
  (other times, by email appt)
  Email is the best way to reach me
- TA: Paul W. Schermerhorn (pscherm1@cse.nd.edu)
- Course Web: http://www.cse.nd.edu/courses/cse498n/www/
- Mailing list:
- Review forum: webct.nd.edu



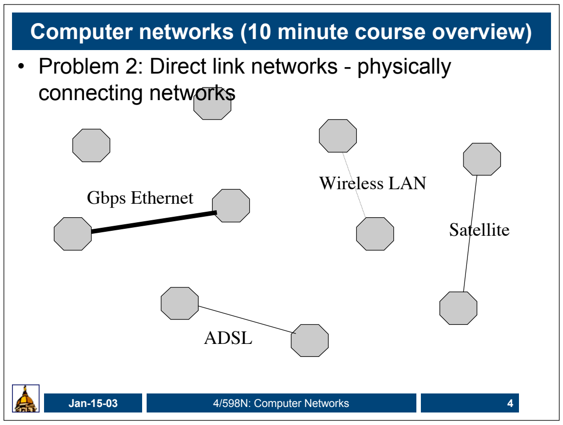
Computer networks (10 minute course overview)

- Goal: Connect all these nodes



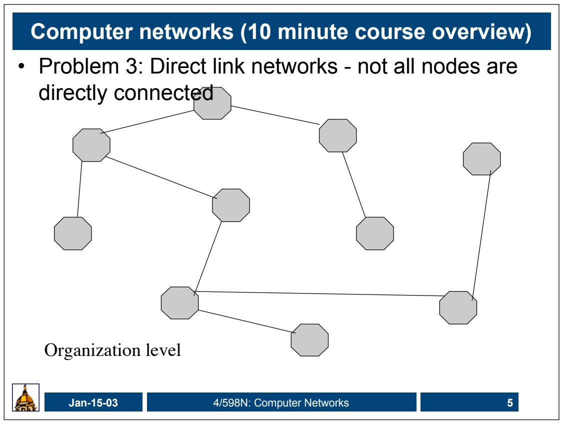
Computer networks (10 minute course overview)

- Problem 1: Network applications



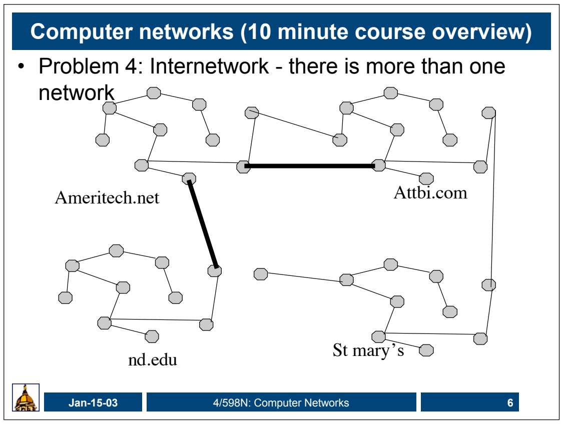
Computer networks (10 minute course overview)

- Problem 2: Direct link networks - physically connecting networks



Computer networks (10 minute course overview)

- Problem 3: Direct link networks - not all nodes are directly connected

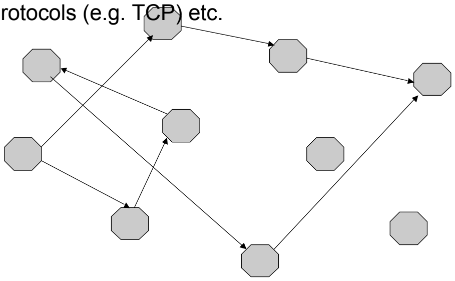


Computer networks (10 minute course overview)

- Problem 4: Internetwork - there is more than one network

Computer networks (10 minute course overview)

- Goal: Congestion control, Reliable. End-to-end protocols (e.g. TCP) etc.



Computer networks (10 minute course overview)

- Rest:
  - Wireless and ad-hoc networks
  - Multicast and QoS - Guest lecture by Prof. Striegel
  - TCP
  - Internet Performance
  - Active Networks
  - P2P (Napster, Gnutella etc) and CDN's (Akamai etc)
  - Security - Guest lecture by Prof. Freeland
  - Network performance and modeling
  - Future directions

Course Goals

- Cover core as well as newer networking technologies
- Goal is to cover as much breadth rather than depth
- Lots of interesting topics to cover
  - Feedback if you would like us to discuss some other technology
- As much hands on experience as possible
- Course project and home work projects should help

Outline for today

- Course policies:
  - Course organization and expectation
  - Grading policy, late policy, reevaluation policy
  - Academic honesty
- Assignment I:
  - P2P Systems
- How to read and write systems papers

Course Organization

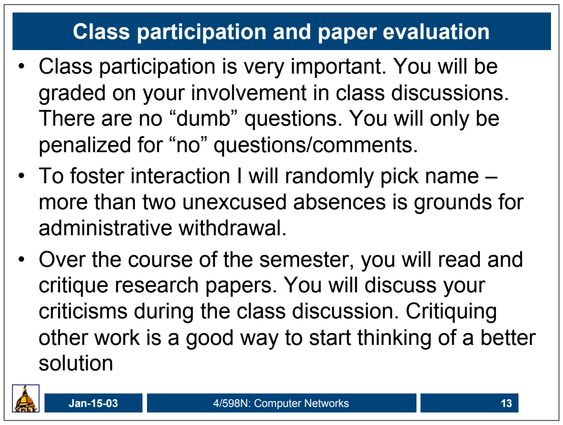
We will follow the course text for the first half with research papers for the second half.

I will augment the discussions with topical research papers

I encourage open discussion about the technologies

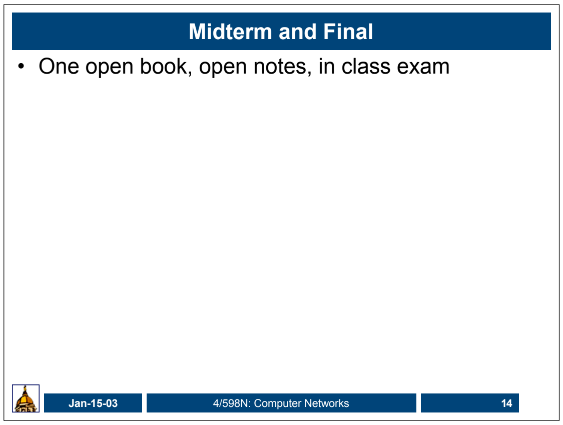
Grade distribution

- Class participation – 4 %
- Paper evaluation – 4 %
- Midterm – 15%
- Final – 15%
- Homework projects – 20% (2 x 10%)
- Take home assignments – 20% (2 x 10%)
- Course project - 22%



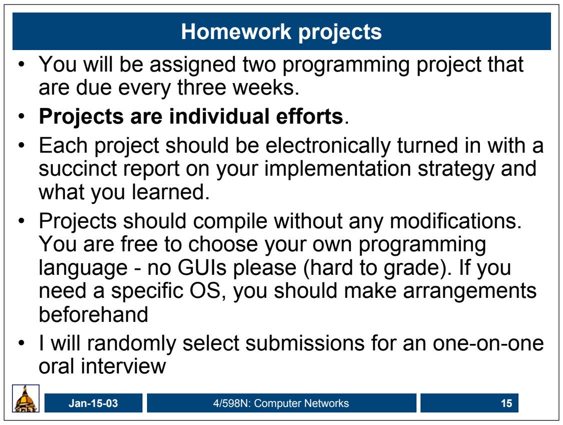
Class participation and paper evaluation

- Class participation is very important. You will be graded on your involvement in class discussions. There are no “dumb” questions. You will only be penalized for “no” questions/comments.
- To foster interaction I will randomly pick name – more than two unexcused absences is grounds for administrative withdrawal.
- Over the course of the semester, you will read and critique research papers. You will discuss your criticisms during the class discussion. Critiquing other work is a good way to start thinking of a better solution



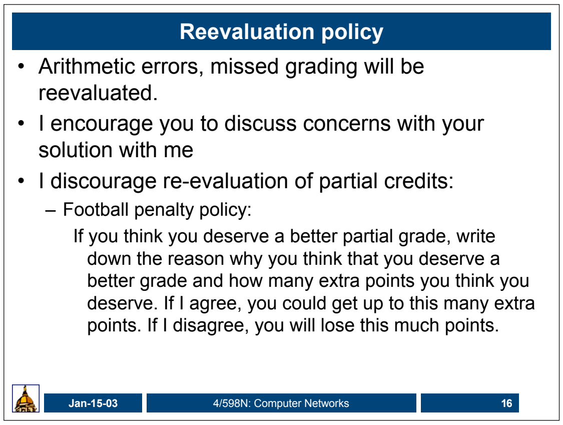
Midterm and Final

- One open book, open notes, in class exam



Homework projects

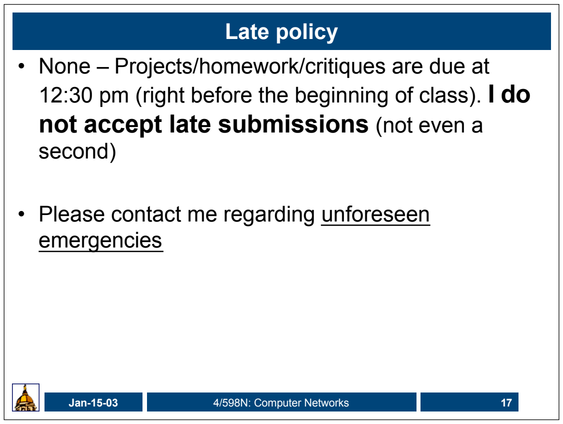
- You will be assigned two programming project that are due every three weeks.
- **Projects are individual efforts**.
- Each project should be electronically turned in with a succinct report on your implementation strategy and what you learned.
- Projects should compile without any modifications. You are free to choose your own programming language - no GUIs please (hard to grade). If you need a specific OS, you should make arrangements beforehand
- I will randomly select submissions for an one-on-one oral interview



Reevaluation policy

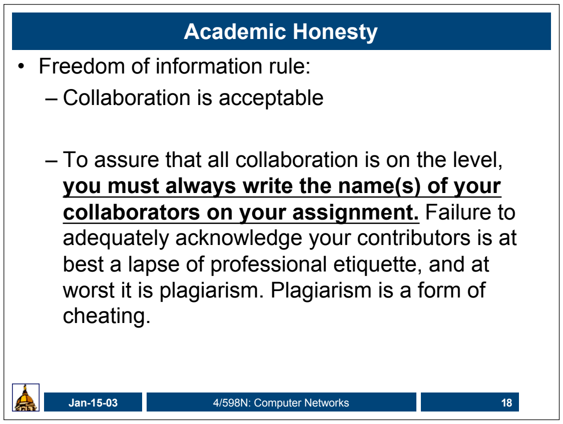
- Arithmetic errors, missed grading will be reevaluated.
- I encourage you to discuss concerns with your solution with me
- I discourage re-evaluation of partial credits:
  - Football penalty policy:

    If you think you deserve a better partial grade, write down the reason why you think that you deserve a better grade and how many extra points you think you deserve. If I agree, you could get up to this many extra points. If I disagree, you will lose this much points.



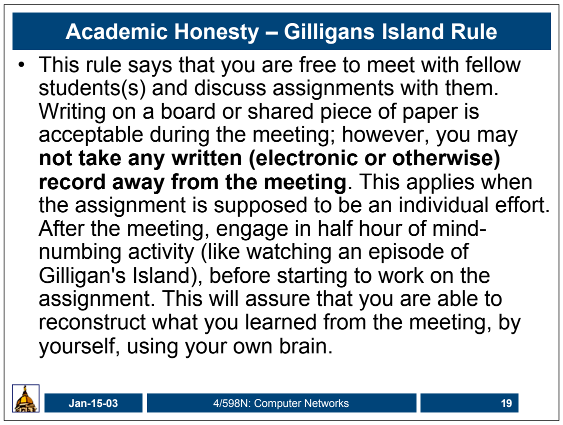
Late policy

- None – Projects/homework/critiques are due at 12:30 pm (right before the beginning of class). **I do not accept late submissions** (not even a second)
- Please contact me regarding unforeseen emergencies



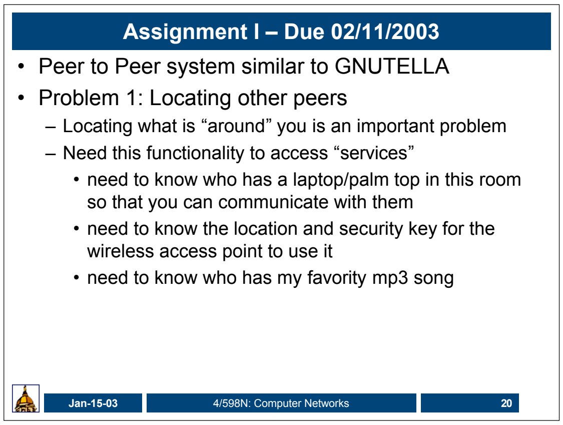
Academic Honesty

- Freedom of information rule:
  - Collaboration is acceptable
  - To assure that all collaboration is on the level, **you must always write the name(s) of your collaborators on your assignment.** Failure to adequately acknowledge your contributors is at best a lapse of professional etiquette, and at worst it is plagiarism. Plagiarism is a form of cheating.



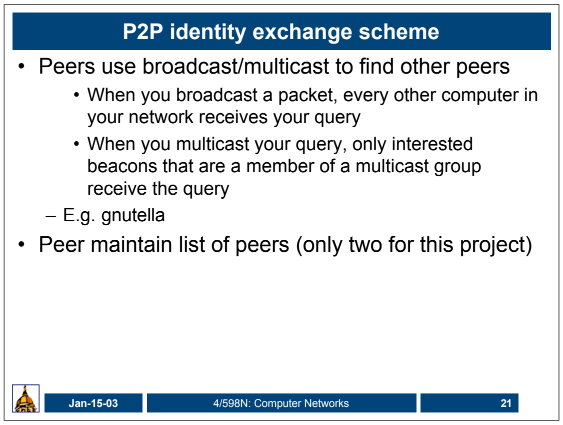
Academic Honesty – Gilligans Island Rule

- This rule says that you are free to meet with fellow students(s) and discuss assignments with them. Writing on a board or shared piece of paper is acceptable during the meeting; however, you may **not take any written (electronic or otherwise) record away from the meeting**. This applies when the assignment is supposed to be an individual effort. After the meeting, engage in half hour of mind-numbing activity (like watching an episode of Gilligan's Island), before starting to work on the assignment. This will assure that you are able to reconstruct what you learned from the meeting, by yourself, using your own brain.



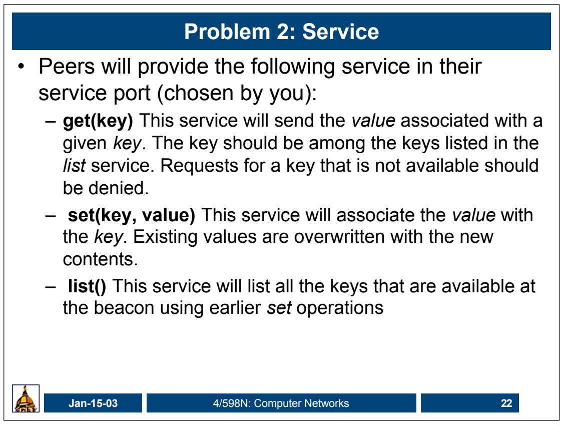
Assignment I – Due 02/11/2003

- Peer to Peer system similar to GNUTELLA
- Problem 1: Locating other peers
  - Locating what is "around" you is an important problem
  - Need this functionality to access "services"
    - need to know who has a laptop/palm top in this room so that you can communicate with them
    - need to know the location and security key for the wireless access point to use it
    - need to know who has my favority mp3 song



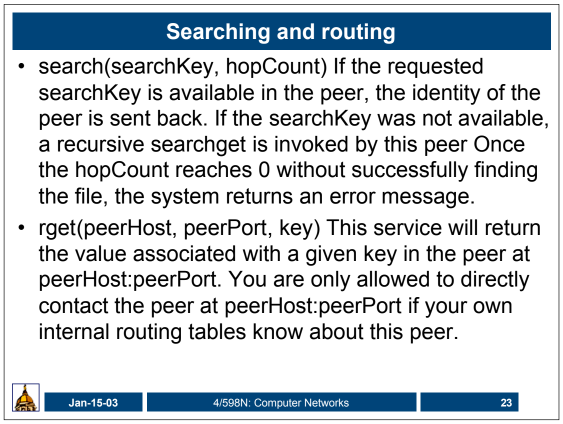
P2P identity exchange scheme

- Peers use broadcast/multicast to find other peers
    - When you broadcast a packet, every other computer in your network receives your query
    - When you multicast your query, only interested beacons that are a member of a multicast group receive the query
  - E.g. gnutella
- Peer maintain list of peers (only two for this project)



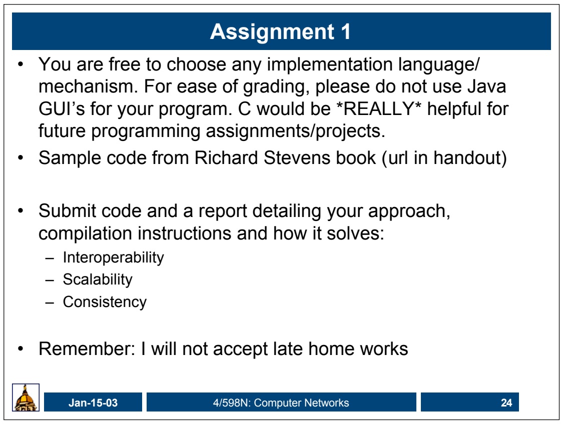
Problem 2: Service

- Peers will provide the following service in their service port (chosen by you):
  - **get(key)** This service will send the *value* associated with a given *key*. The key should be among the keys listed in the *list* service. Requests for a key that is not available should be denied.
  - **set(key, value)** This service will associate the *value* with the *key*. Existing values are overwritten with the new contents.
  - **list()** This service will list all the keys that are available at the beacon using earlier *set* operations



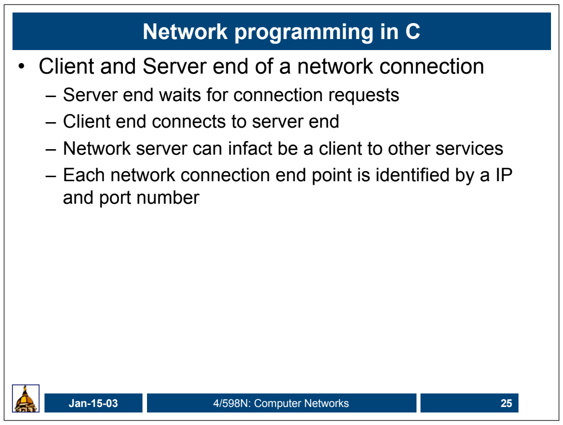
Searching and routing

- search(searchKey, hopCount) If the requested searchKey is available in the peer, the identity of the peer is sent back. If the searchKey was not available, a recursive searchget is invoked by this peer Once the hopCount reaches 0 without successfully finding the file, the system returns an error message.
- rget(peerHost, peerPort, key) This service will return the value associated with a given key in the peer at peerHost:peerPort. You are only allowed to directly contact the peer at peerHost:peerPort if your own internal routing tables know about this peer.



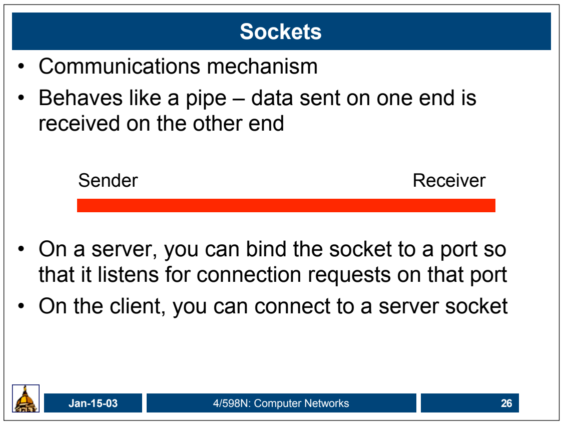
Assignment 1

- You are free to choose any implementation language/ mechanism. For ease of grading, please do not use Java GUI's for your program. C would be *REALLY* helpful for future programming assignments/projects.
- Sample code from Richard Stevens book (url in handout)
- Submit code and a report detailing your approach, compilation instructions and how it solves:
  - Interoperability
  - Scalability
  - Consistency
- Remember: I will not accept late home works



Network programming in C

- Client and Server end of a network connection
  - Server end waits for connection requests
  - Client end connects to server end
  - Network server can infact be a client to other services
  - Each network connection end point is identified by a IP and port number

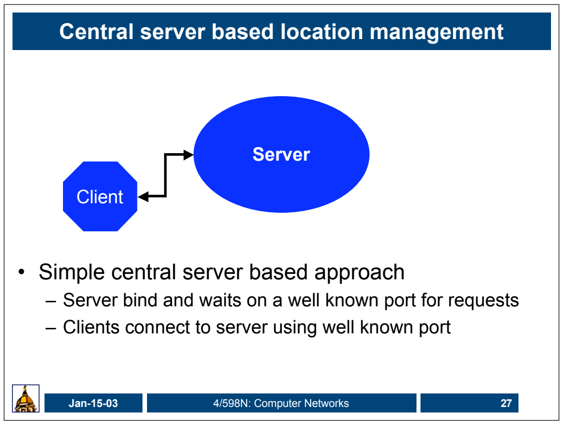


Sockets

- Communications mechanism
- Behaves like a pipe – data sent on one end is received on the other end

Sender
Receiver

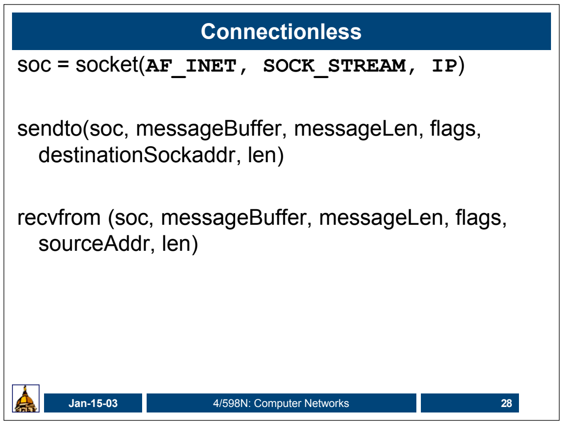
- On a server, you can bind the socket to a port so that it listens for connection requests on that port
- On the client, you can connect to a server socket



Central server based location management

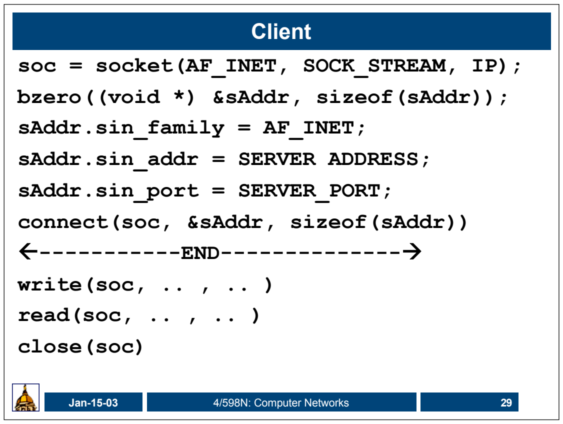
Server
Client

- Simple central server based approach
  - Server bind and waits on a well known port for requests
  - Clients connect to server using well known port



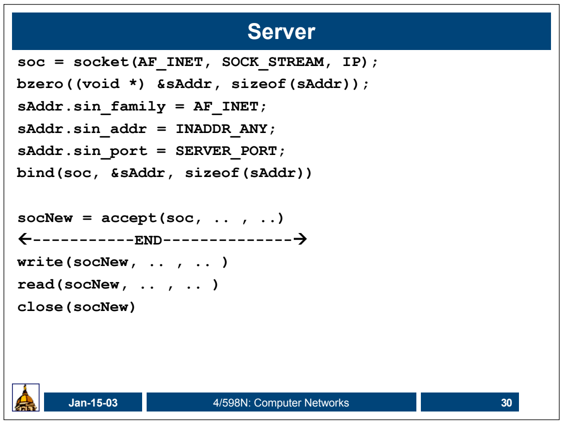
Connectionless

```
soc = socket(AF_INET, SOCK_STREAM, IP)

sendto(soc, messageBuffer, messageLen, flags,
   destinationSockaddr, len)

recvfrom (soc, messageBuffer, messageLen, flags,
   sourceAddr, len)
```



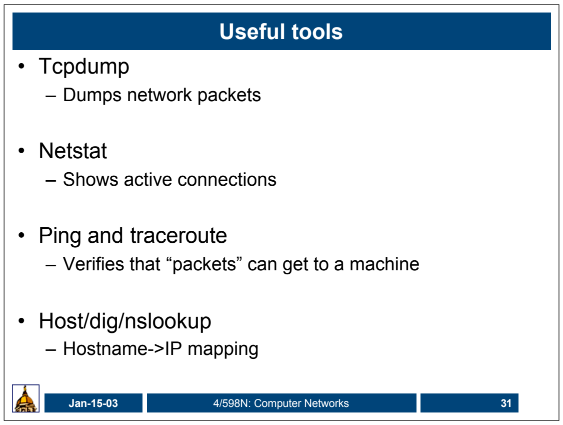
Client

```
soc = socket(AF_INET, SOCK_STREAM, IP);
bzero((void *) &sAddr, sizeof(sAddr));
sAddr.sin_family = AF_INET;
sAddr.sin_addr = SERVER ADDRESS;
sAddr.sin_port = SERVER_PORT;
connect(soc, &sAddr, sizeof(sAddr))
←-----------END--------------→
write(soc, .. , .. )
read(soc, .. , .. )
close(soc)
```



Server

```
soc = socket(AF_INET, SOCK_STREAM, IP);
bzero((void *) &sAddr, sizeof(sAddr));
sAddr.sin_family = AF_INET;
sAddr.sin_addr = INADDR_ANY;
sAddr.sin_port = SERVER_PORT;
bind(soc, &sAddr, sizeof(sAddr))

socNew = accept(soc, .. , ..)
←-----------END--------------→
write(socNew, .. , .. )
read(socNew, .. , .. )
close(socNew)
```



Useful tools

- Tcpdump
  - Dumps network packets
- Netstat
  - Shows active connections
- Ping and traceroute
  - Verifies that “packets” can get to a machine
- Host/dig/nslookup
  - Hostname->IP mapping