61A Lecture 33

Monday, November 25

Announcements

- Homework 10 due Tuesday 11/26 @ 11:59pm
- No lecture on Wednesday 11/27 or Friday 11/29
- No discussion section Wednesday 11/27 through Friday 11/29
  - Lab will be held on Wednesday 11/27
- Recursive art contest entries due Monday 12/2 @ 11:59pm
- Guerrilla section about logic programming coming soon...
- Homework 11 due Thursday 12/5 @ 11:59pm

Addition in Logic

(Demo)

Distributed Computing

Distributed Computing

A **distributed computing application** consists of multiple programs running on multiple computers that together coordinate to perform some task.

- Computation is performed in *parallel* by many computers.
- Information can be *restricted* to certain computers.
- Redundancy and geographic diversity improve *reliability*.

Characteristics of distributed computing:

- Computers are *independent* — they do not share memory.
- Coordination is enabled by *messages* passed across a network.
- Individual programs have differentiating *roles*.

Distributed computing for **large-scale data processing:**

- Databases respond to queries over a network.
- Data sets can be partitioned across multiple machines (next lecture).

Network Messages

Computers communicate via messages: sequences of bytes transmitted over a network.

Messages can serve many purposes:

- **Send data** to another computer
- **Request data** from another computer
- Instruct a program to **call a function** on some arguments.
- **Transfer a program** to be executed by another computer.

Messages conform to a *message protocol* adopted by both the sender (to encode the message) & receiver (to interpret the message).

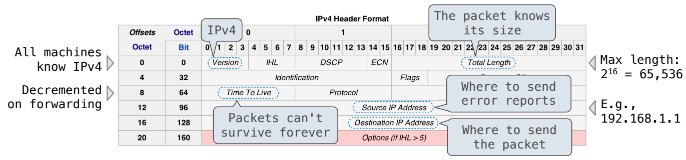
- For example, bits at fixed positions may have fixed meanings.
- Components of a message may be separated by delimiters.
- Protocols are designed to be implemented by many different programming languages on many different types of machines.

Internet Protocol

The Internet Protocol

The Internet Protocol (IP) specifies how to transfer *packets* of data among networks.

- Networks are inherently unreliable at any point.
- The structure of a network is dynamic, not fixed.
- No system exists to monitor or track communications.



Packets are forwarded toward their destination on a best effort basis.

Programs that use IP typically need a policy for handling lost packets.

http://en.wikipedia.org/wiki/IPv4

Transmission Control Protocol

Transmission Control Protocol

The design of the **Internet Protocol** (IPv4) imposes constraints:

- Packets are limited to 65,535 bytes each.
- Packets may arrive in a different order than they were sent.
- Packets may be duplicated or lost.

The **Transmission Control Protocol** (TCP) improves reliability:

- Ordered, reliable transmission of arbitrary byte streams.
- Implemented using the IP. Every TCP connection involves sending IP packets.
- Each packet in a TCP session has a sequence number:
  - The receiver can correctly order packets that arrive out of order.
  - The receiver can ignore duplicate packets.
- All received packets are acknowledged; both parties know that transmission succeeded.
- Packets that aren't acknowledged are sent repeatedly.

The **socket** module in Python implements the TCP.

TCP Handshakes

All TCP connections begin with a sequence of messages called a "handshake" which verifies that communication is possible.

"Can you hear me now?" Let's design a handshake protocol.

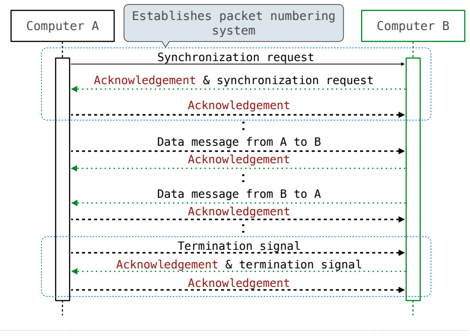
Handshake Goals:

- Computer A knows that it can *send data to and receive data from* Computer B.
- Computer B knows that it can *send data to and receive data from* Computer A.
- Lots of separate connections can exist without any confusion.
- The number of required messages is minimized.

Communication Rules:

- Computer A can send an initial message to Computer B requesting a new connection.
- Computer B can respond to messages from Computer A.
- Computer A can respond to messages from Computer B.

Message Sequence of a TCP Connection



Client/Server Architecture

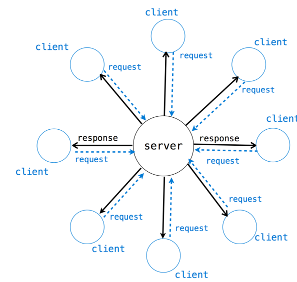
The Client/Server Architecture

One server provides information to multiple clients through *request* and *response* messages.

Server role: Respond to service requests with requested information.

Client role: Request information and make use of the response.

Abstraction: The client knows *what service* a server provides, but not *how* it is provided.



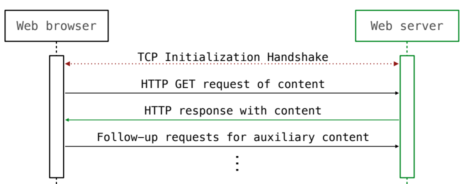
Client/Server Example: The World Wide Web

The **client** is a web browser (e.g., Firefox):

- Request content for a location.
- Interpret the content for the user.

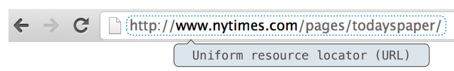
The **server** is a web server:

- Interpret requests and respond with content.



The Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is a protocol designed to implement a Client/Server architecture.



Browser issues a GET request to a server at www.nytimes.com for the content (resource) at location "pages/todayspaper".

Server response contains more than just the resource itself:

- Status code, e.g. **200** OK, **404** Not Found, **403** Forbidden, etc.
- Date of response; type of server responding
- Last-modified time of the resource
- Type of content and length of content

Properties of a Client/Server Architecture

Benefits:

- Creates a separation of concerns among components.
- Enforces an abstraction barrier between client and server.
- A centralized server can reuse computation across clients.

Liabilities:

- A single point of failure: the server.
- Computing resources become scarce when demand increases.

Common use cases:

- Databases — The database serves responses to query requests.
- Open Graphics Library (OpenGL) — A graphics processing unit (GPU) serves images to a central processing unit (CPU).
- Internet file and resource transfer: HTTP, FTP, email, etc.

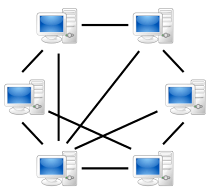
Peer-to-Peer Architecture

The Peer-to-Peer Architecture

All participants in a distributed application contribute computational resources: processing, storage, and network capacity.

Messages are relayed through a network of participants.

Each participant has only partial knowledge of the network.

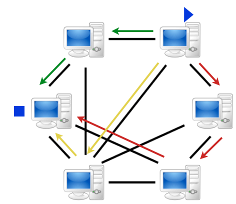


http://en.wikipedia.org/wiki/File:P2P-network.svg

Network Structure Concerns

Some data transfers on the Internet are faster than others.

The time required to transfer a message through a peer-to-peer network depends on the route chosen.



http://en.wikipedia.org/wiki/File:P2P-network.svg

Example: Skype

Skype is a Voice Over IP (VOIP) system that uses a hybrid peer-to-peer architecture.

Login & contacts are handled via a centralized server.

Conversations between two computers that cannot send messages to each other directly are relayed through *supernodes*.

Any Skype client with its own IP address may be a supernode.