Project Overview & Network Programming Guide

18-441/741
Spring 2018

Networked System Design

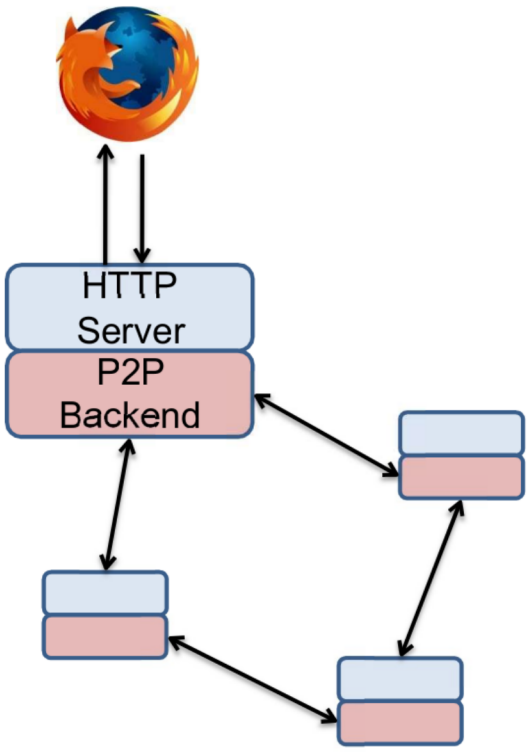
Give me the place to stand, and I shall move the earth.

Give me a data stream of bits and I will change the world.

The main difference between a software system and a networked system are the parameters of optimization and metrics of evaluation

	Software Systems	Networked Systems
Evaluation Metrics	Space and time complexity, modularity	Bandwidth utilized, Latency, Scalability
Optimization parameters	CPU/GPU utilization, memory management	Network Capacity, Server Quality, Up-time
Interoperability	Can work on one type of OS or hardware	Must be interoperable across platforms

Project Overview

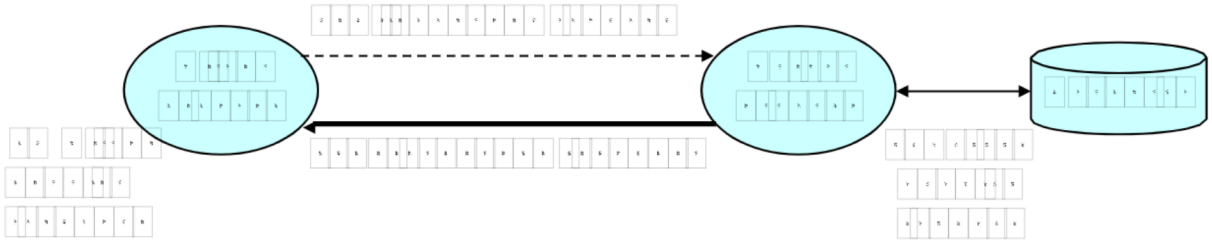


- P2P Video-on-Demand
  - Distributed "YouTube"
- Eventually grows to a full peer-to-peer VoD system
- Apply network concepts
  - Socket/Server programming
  - Congestion control
  - Failure detection/recovery
  - Distributed hash

Network Programming

A Client-Server Transaction

- Most network applications are based on the client-server model:
  - A server process and one or more of client processes.
  - Server manages some resource.
  - Server provides service by manipulating resource for clients.



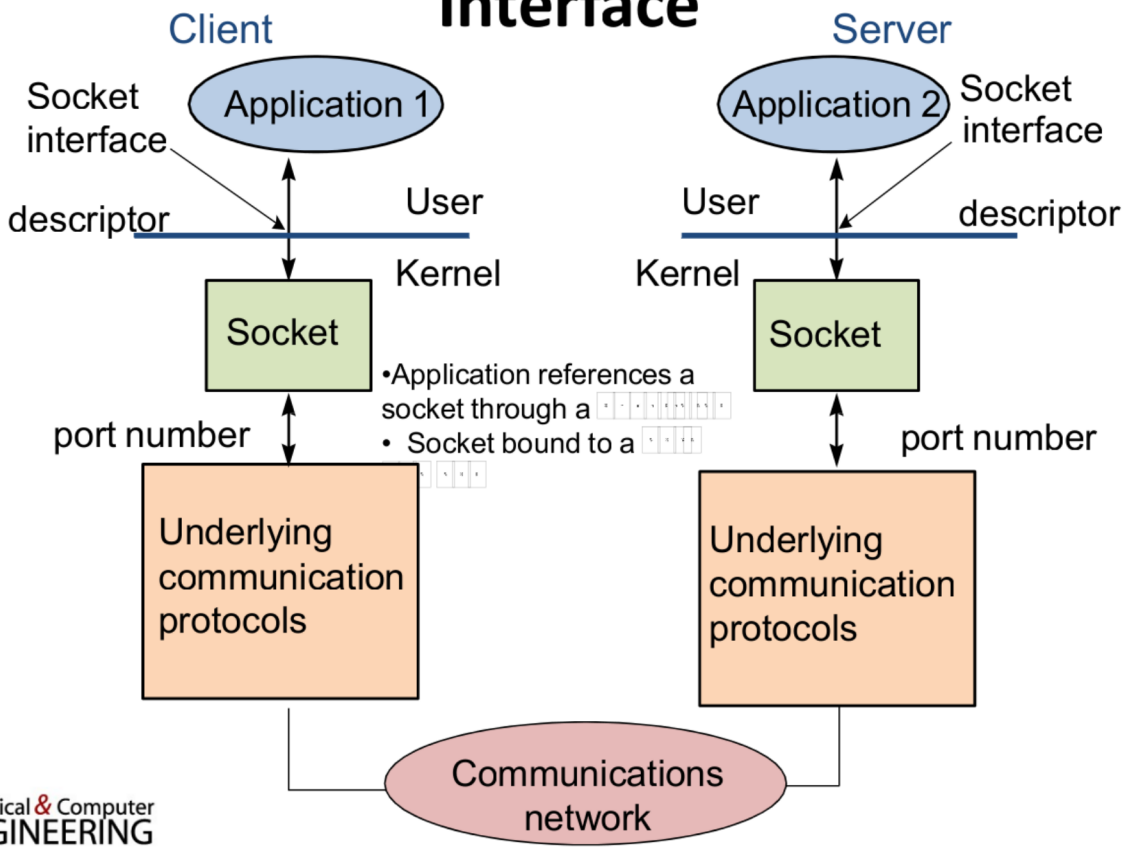
The Socket Interface

- What is a **socket?**
  - A descriptor that lets an application read/write from/to the network
  - Similar abstraction for network I/O as file I/O
  - Clients and servers communicate by reading/writing from/to socket descriptors

Socket API

- API (Application Programming Interface)
  - Provides a standard set of functions that can be called by applications
- Berkeley UNIX Sockets API (C)
  - Abstraction for applications to send & receive data
  - Applications create sockets that "plug into" network
  - Applications write/read to/from sockets
  - Implemented in the kernel
  - Facilitates development of network applications
  - Hides details of underlying protocols & mechanisms
- Also in Windows, Linux, and other OS's
- Socket API is also available in Java (java.net.socket)

Communications through Socket Interface

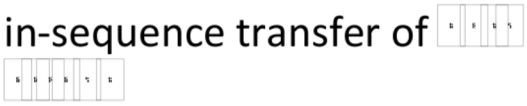


Internet Connections

- Clients and servers communicate by sending streams of bytes over connections.
- Socket address is identified by an **IPaddress:port** pair.
- A **port** is a 16-bit unsigned integer identifying a process (ephemeral, not physical port.)
- Ports below 1024 are reserved for standard protocols.

Stream mode of service

Connection-oriented

- First, setup connection between two peer application processes
- Then, reliable bidirectional in-sequence transfer of 
- Multiple write/read between peer processes
- Finally, connection release
- Uses TCP

Connectionless (Datagram)

- Immediate transfer of one block of information
- No setup overhead & delay
- Destination address with each block
- Send/receive to/from multiple peer processes
- Best-effort service only
  - Possible out-of-order
  - Possible loss
- Uses UDP

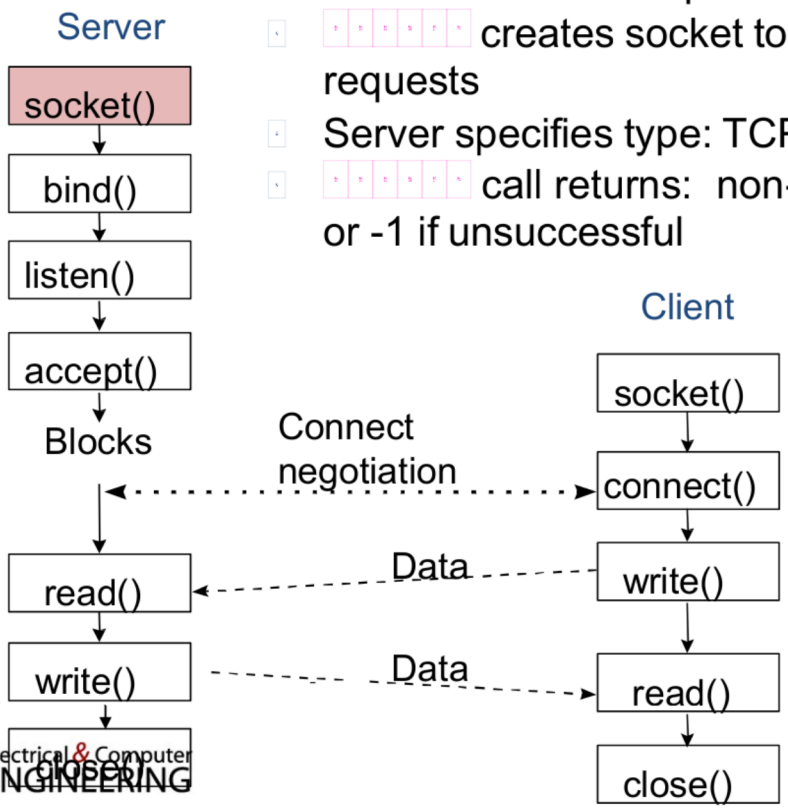
Client & Server Differences

- Server
  - Specifies well-known port # when creating socket
  - May have multiple IP addresses (net interfaces)
  - Waits passively for client requests
- Client
  - Assigned ephemeral port #
  - Initiates communications with server
  - Needs to know server's IP address & port #
    - DNS for URL & server well-known port #
  - Server learns client's address & port #
- Peer-to-Peer
  - A process performs as both server and client

Socket Calls for Connection-Oriented Mode

Server does Passive Open

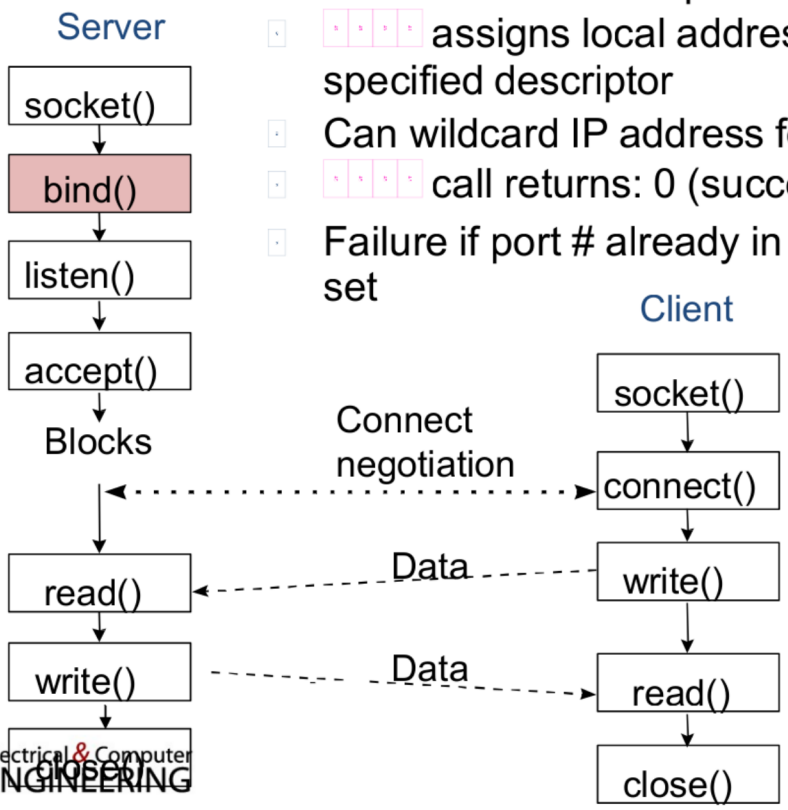
- [illegible] creates socket to [illegible] for connection requests
- Server specifies type: TCP (stream)
- [illegible] call returns: non-negative integer [illegible]; or -1 if unsuccessful



Socket Calls for Connection-Oriented Mode

Server does Passive Open

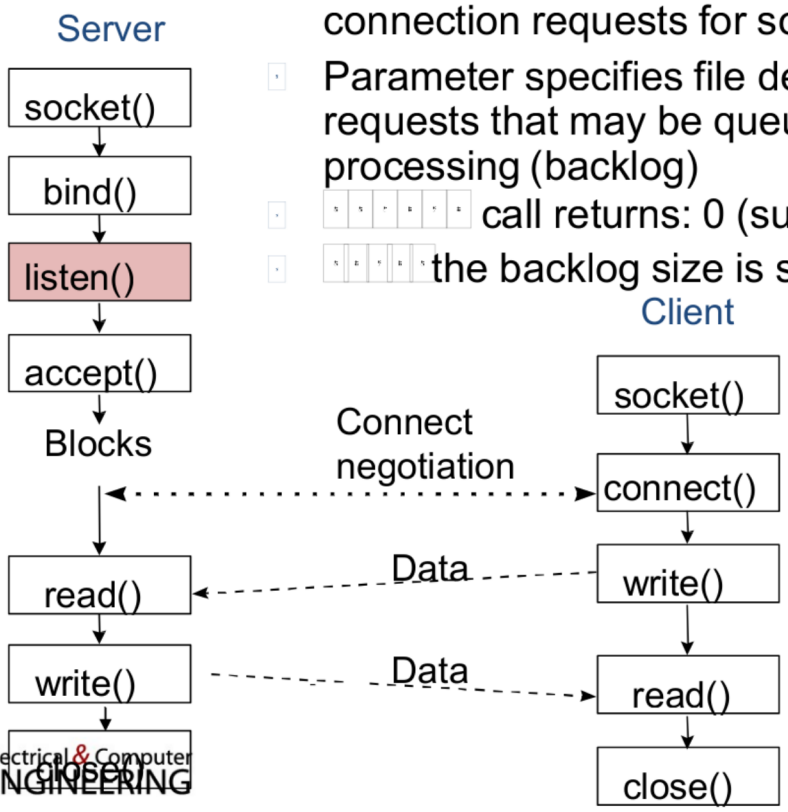
- bind() assigns local address & port # to socket with specified descriptor
- Can wildcard IP address for multiple net interfaces
- bind() call returns: 0 (success); or -1 (failure)
- Failure if port # already in use or if reuse option not set



Socket Calls for Connection-Oriented Mode

Server does Passive Open

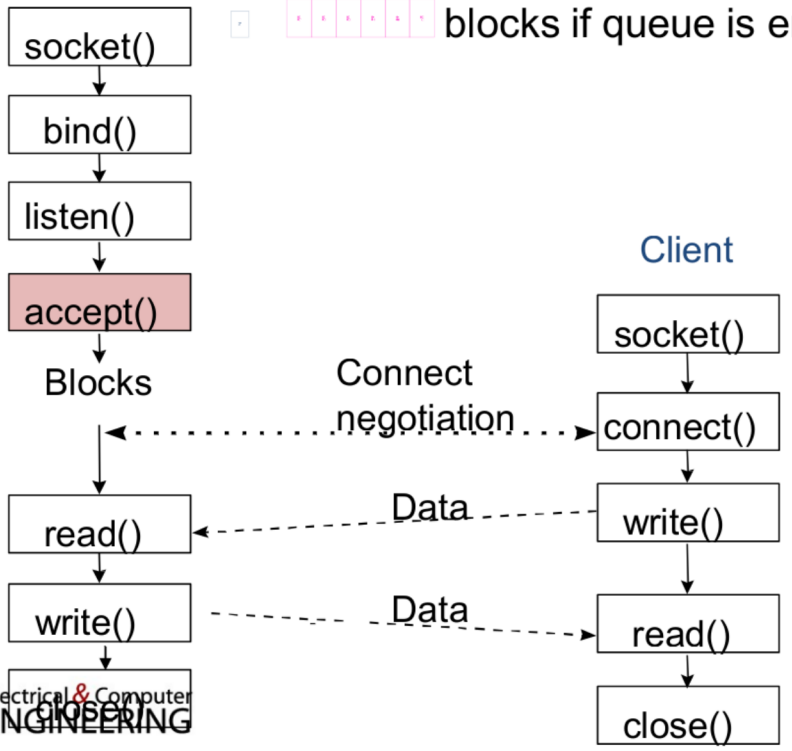
- [illegible] indicates TCP readiness to receive connection requests for socket with given descriptor
- Parameter specifies file descriptor and max number of requests that may be queued while waiting for processing (backlog)
- [illegible] call returns: 0 (success); or -1 (failure)
- [illegible] the backlog size is specified thru [illegible]



Socket Calls for Connection-Oriented Mode

Server does Passive Open

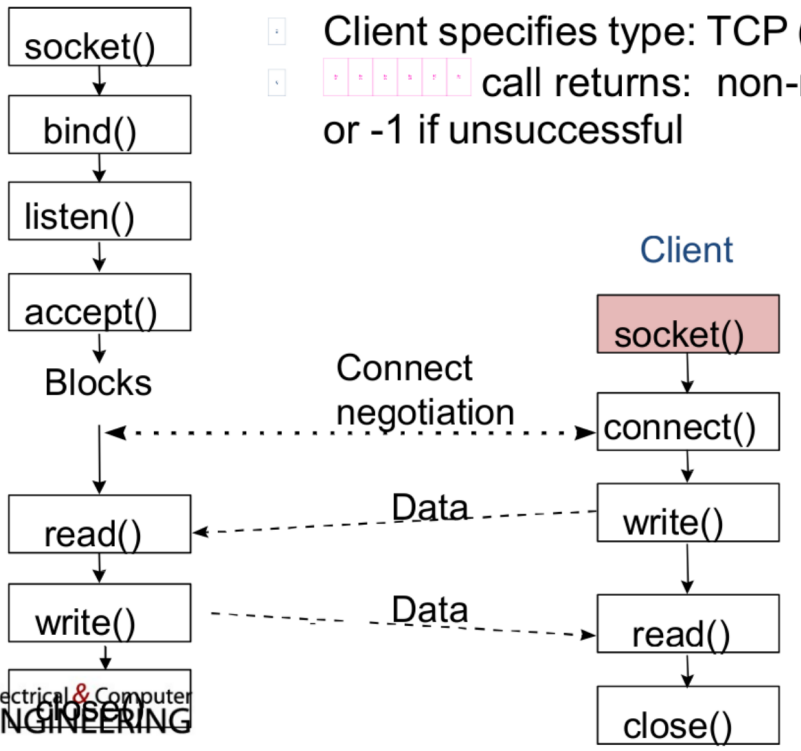
- Server calls [illegible] to accept incoming requests
- [illegible] blocks if queue is empty



Socket Calls for Connection-Oriented Mode

Client does Active Open

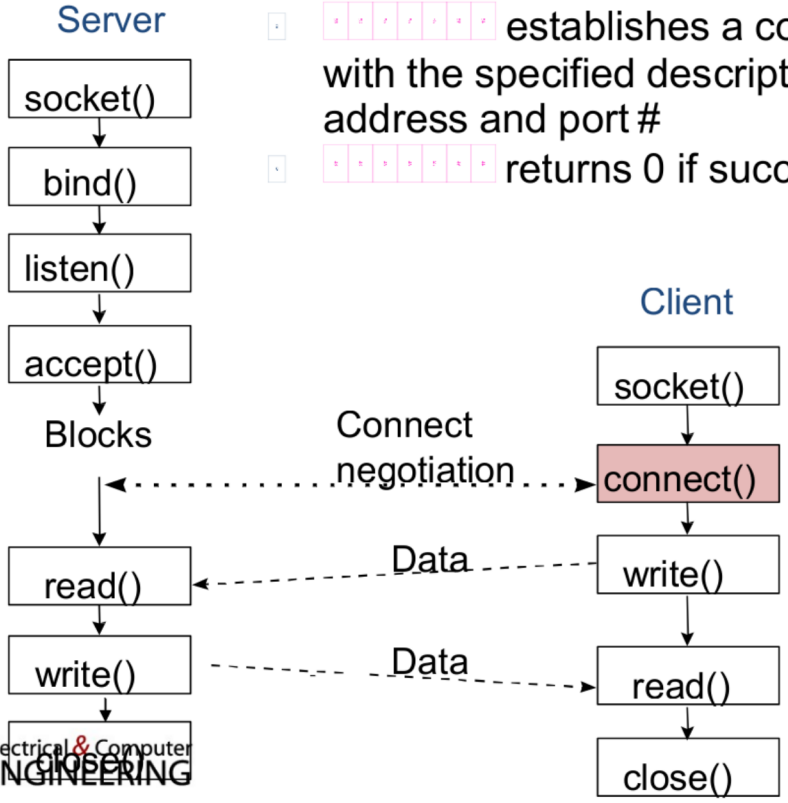
- creates socket to connect to server
- Client specifies type: TCP (stream)
- call returns: non-negative integer ; or -1 if unsuccessful



Socket Calls for Connection-Oriented Mode

Client does Active Open

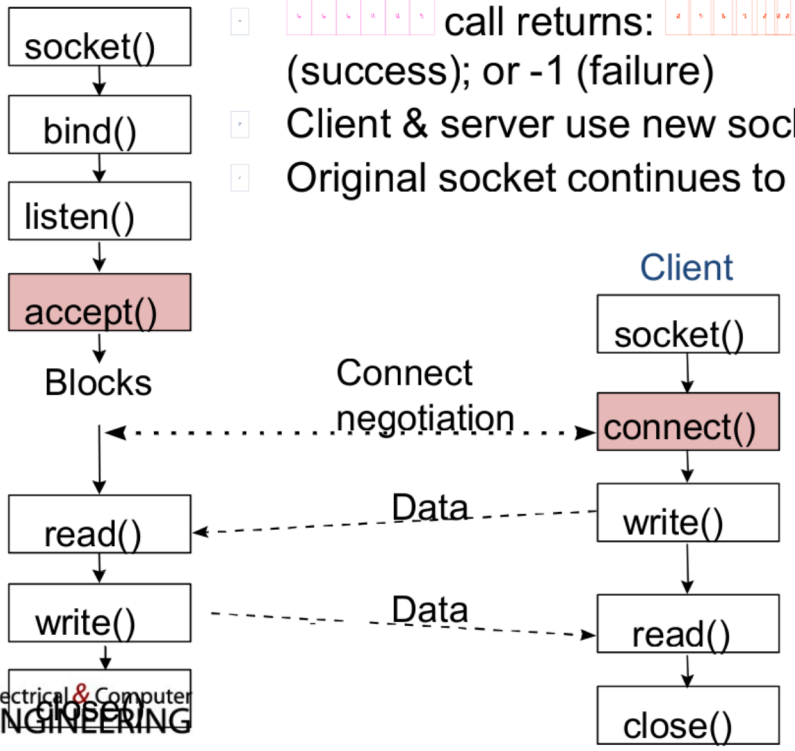
- [redacted] establishes a connection on the local socket with the specified descriptor to the specified remote address and port #
- [redacted] returns 0 if successful; -1 if unsuccessful



Note: [redacted] initiates TCP three-way handshake

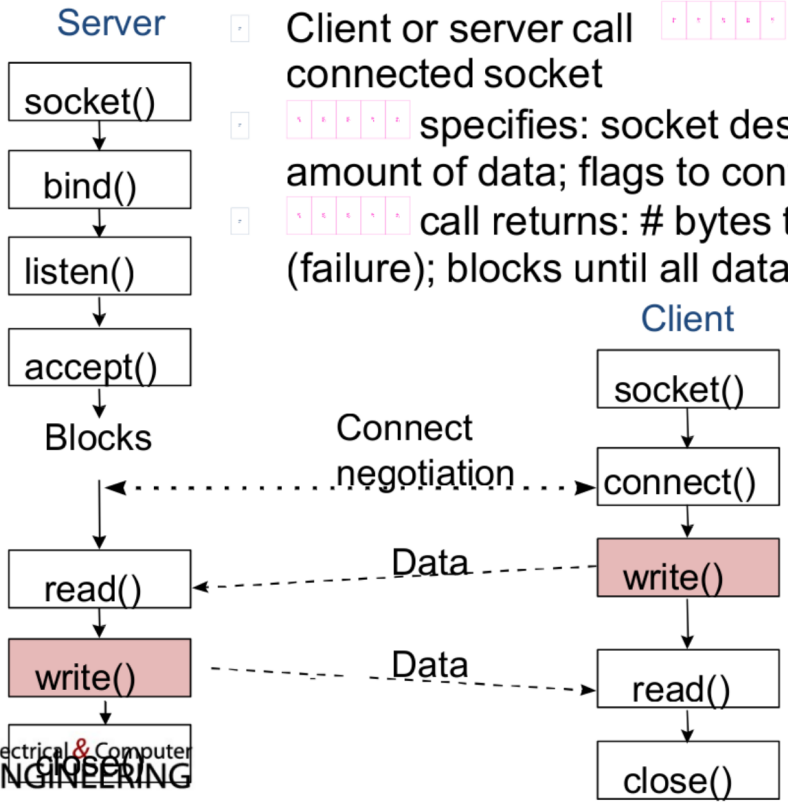
Socket Calls for Connection-Oriented Mode

- [illegible] wakes with incoming connection request
- [illegible] fills client address & port # into address structure
- [illegible] call returns: [illegible] (success); or -1 (failure)
- Client & server use new socket for data transfer
- Original socket continues to listen for new requests



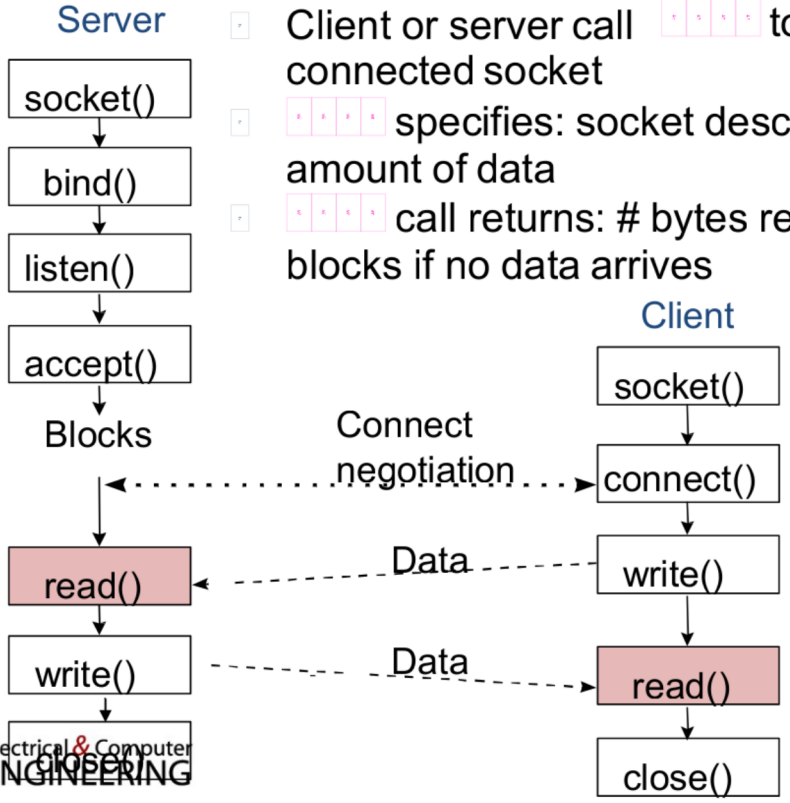
Socket Calls for Connection-Oriented Mode

- Client or server call [illegible] to transmit data into a connected socket
- [illegible] specifies: socket descriptor; pointer to a buffer; amount of data; flags to control transmission behavior
- [illegible] call returns: # bytes transferred (success); or -1 (failure); blocks until all data transferred



Socket Calls for Connection-Oriented Mode

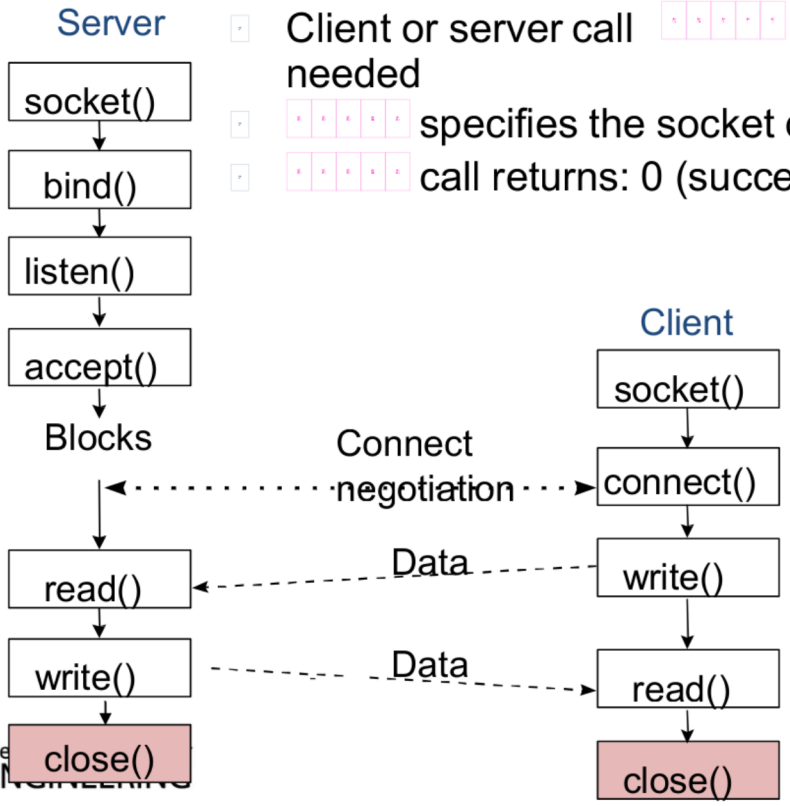
- Client or server call [illegible] to receive data from a connected socket
- [illegible] specifies: socket descriptor; pointer to a buffer; amount of data
- [illegible] call returns: # bytes read (success); or -1 (failure); blocks if no data arrives



Note: [illegible] and [illegible] can be called multiple times to transfer byte streams in both directions

Socket Calls for Connection-Oriented Mode

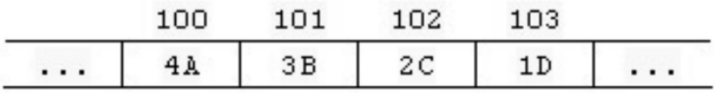
- Client or server call [illegible] when socket is no longer needed
- [illegible] specifies the socket descriptor
- [illegible] call returns: 0 (success); or -1 (failure)



Note: [illegible] initiates TCP graceful close sequence

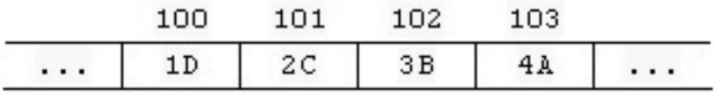
Byte-Ordering

- Consider a hexadecimal **4A3B2C1D** at address 100. the bytes could be stored within the address range 100 through 103 in the following order:
- **Big-endian ("big end first")**



  - the **most significant byte** (**msb**, 4A) is stored at the lowest address.
  - Used by Motorola/SPARC and **network devices**.

- **Little-endian ("little end first")**



  - The **least significant byte** byte (**lsb,** 1D) is stored at the lowest address
  - Used by Intel x86, DEC VAX

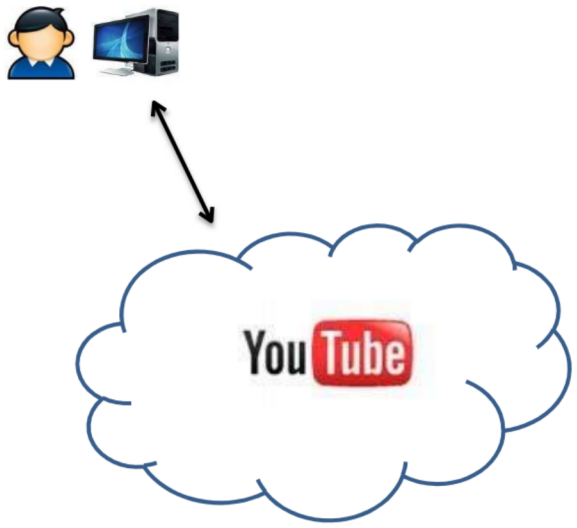
- Endianness does ' ' ' denote what the value ' ' ' ' with when stored in memory, but rather ' ' ' ' ' ' ' it begins with.

Demo (using Python)

Project Overview

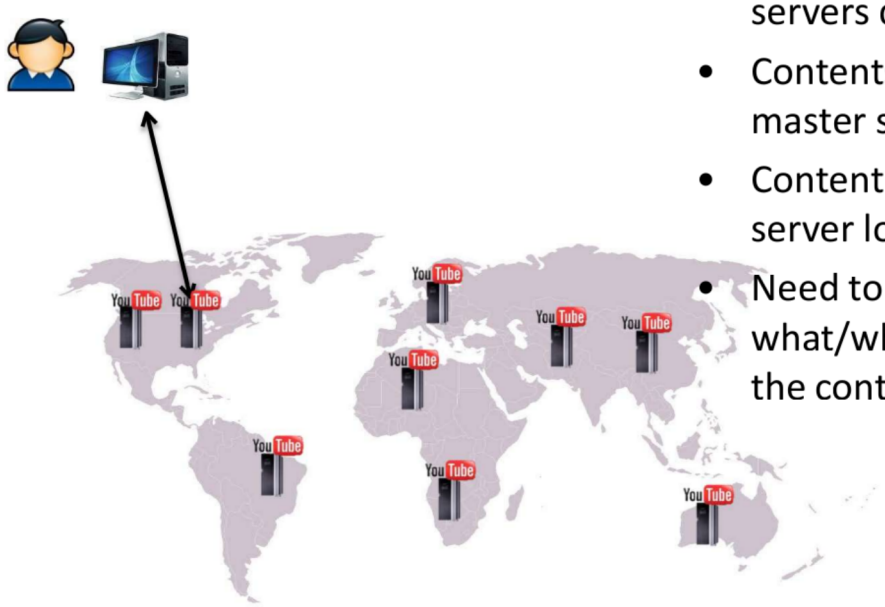
Electrical & Computer ENGINEERING

Video-on-demand



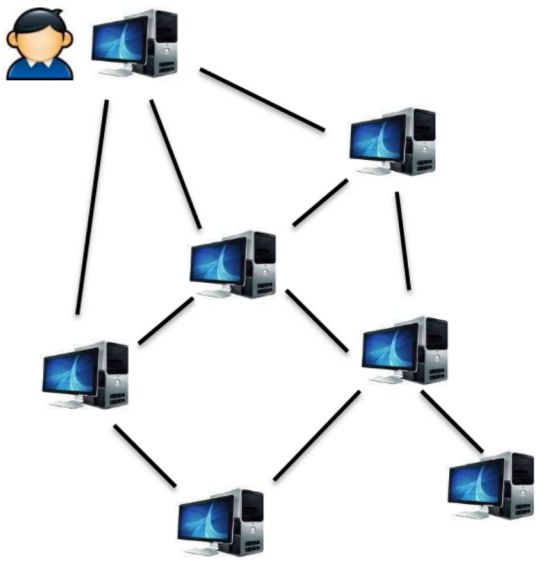
- Video on demand
  - View what you want whenever you want
  - Usually not 'live' stream
- Content serves from a single provider
  - Not necessarily from the same server

Content Distribution Network



- Service providers have many servers distributed geographically
- Contents are duplicated from the master server to other locations
- Content is served from a **single** server located near the client
- Need to be concerned about what/where/when to duplicate the contents

P2P Video-on-demand



- Peer-to-Peer network
  - All nodes provide resource (bandwidth, content)
  - Many nodes may share the same(popular) contents
- P2P Video on demand
  - Need to be able to query and locate the desired content
  - Clients may concurrently send/retrieve content to/from multiple peers
- Performance consideration
  - Content search time
  - Content retrieval time
  - Content "ready-to-view" time
  - Network bandwidth required

Serving Video over IP

Streaming Media

- The content is constantly received and processed by the client
- Small client buffer needed (client buffer may not hold the entire content)
- Protocols provide facility for jitter compensation / handle packet loss / synchronization
- Utilize multiple protocols
  - Transport:
    - Real-time Transport Protocol (RTP)
  - Control:
    - Real Time Control Protocol (RTCP)
    - Real-time Streaming Protocol (RTSP)
    - Session-initiation Protocol (SIP)
- Example
  - Flash Media Server
  - Quicktime Streaming Server
  - VLC

Progressive Download (Pseudo Streaming)

- Allows media playback while downloading the content
- Client uses content metadata to enable early playback
- Client buffer used to store the entire content
- Protocol
  - Hypertext Transfer Protocol (HTTP)
- Example
  - Youtube / Flash
  - JPEG (back when we use modem)
  - HTML 5 audio/video (use in our project)