***Chapter 2*Client vs P2P** Servers are always on hosts. And all clients communicate with the server. In P2P Architecture, there is no always on host. Clients connect directly to one another.   
**Sockets** processes send and receive message to/from its socket. A socket is between the application layer and the transport layer.  
**TCP/UDP** TCP offers reliable transport as well as flow control, congestion control but does not provide timing, and thru put guarantee, nor security. UDP is unreliable, and does not offer anything. It is barebones shoot and forget transport.  
**Persistent vs Non-persistent HTTP Connections** Non at most one object can be sent over TCP connection. The connection is then closed. Multi Objects means multi connections. Persistent is polar opposite. Multiple objects can be sent. The Overhead for connections is lost in persistent. One connection. Many objects.  
**Web Caches** In web caches, the ISP or local server contains the web cache. This reduces response time on the end user. And reduces traffic on the institution’s access link.  
**FTP** Port 21. USER,PASS,LIST,RETR,STOR {331 OK, 125 connect already open,425 can’t open connect. 452 error writing}  
**SMTP** TCP. Port 25. Handshake, transfer,close. ASCII Text. 7-bit. Mail Serv to Mail Serv. HELO,MAIL FROM, RCPT TO, DATA, QUIT single period CRLF.CRLF ends the message. SMTP Pushes.  **POP** Post Office Protocol. Download and delete. No Stateless, and no organization of the message. List,retr,dele,quit **IMAP** Internet Mail Access Protocol. Everything POP doesn’t have.   
**DNS** Domain Name System. Human Readable to IP. .com/.org/.net all have a “DB” that is used for lookup. A = hostname,value=IP;CNAME=name is canonical; NS=name is the domain, value is hostname; MX=mailerserver associated with name  
**P2P** D\_c-s >= max {NF/u\_s,F/d\_min} where N is file copies, d\_min is min client d/l rate and F/d\_min is max client download time. With P2P, we are fast. We are only as slow if little clients are connected.   
**Socket Programming** A socket is the “door” or connector from the application level, process, to the transport layer. UDP uses datagrams, and is a fire and forget style message sending. TCP is reliable, stream open connection messaging system. UDP : sendto(), SOCK\_DGRAM, recvfrom(); TCP : SOCK\_STREAM, send(), recv(),listen()