develop software: scalable, fault-tolerant, evolvable, secure
How to build(top to bottom): systems, top of abstractions, software, hardware
point-to-point&multiple access, switched network&interconnection of networks
latency=propagation+transmit+queue
overhead:# secs for CPU to put message on wire
Bandwidth(available over link) vs. **throughput**(available to application)
protocol: explicit and implicit conventions for how to communicate
specified protocol: Prose/BNF, State transition diagrams, message sequence diagram, packet formats
thin waist: physical, datalink, network(IP), transport, application
frames: break down a stream of bits into smaller, digestible chunks (header+payload+trailer)
reliable transmission: simple idea: ARQ(Automatic Repeat Request))
flow control: tell sender how much buffer left at receiver
forwarding: move packet form input link to the appropriate output link, purely local computation, must go be very fast.   spanning tree
routing: make sure that the next hop actually leads to the destination, global decision: distributed computation and communication, can go slower
link state: tell everyone you know about neighbor
distance vector: tell neighbor you know about everyone
address: network+subnet+host (prefix = network+subnet, subnet specified by netmask)
!!!CIDR(classless inter-domain routing) address:network+host(network is prefix)
TCP,UDP use ports: below 1024, well-known port numbers, above,OS-assignend temporary..
UDP: unreliable, connectionless
UDP: data, UDP header, IP pseudoheader(history)
TCP: bi-directional bytestream, connection-oriented, flow control(prevent sender from over-running receiver buffer), congestion control(…network capacity)
SQS: SendQ size RQS: RecvQ size
n>SQS: blocks until (n-SQS) bytes xfered to RecvQ
n>(SQS+RQS): blocks until receiver calls recv() enough to read in n-(SQS+RQS) bytes.
WEB/HTTP overview: Documents link to other documents(HTML files) +HTTP(protocol for retrieving HTML files from server)+implemented in servers+ clients(Chrome, Safari…)
HTTP overview: test oriented protocol, request/response protocol
<CRLF>carriage-return-line feed
##HTTP
##HTTP REQUESTS: operation(method), web page, version of HTTP
##OPTIONAL HEADERS: request headers: Text-based, key and value separated by a colon
exp. Host: www.cs.ucsd.edu
##START RESPONSE: version of HTTP, tree-digit status code, Test string (reason why this resp)
HTTP response code: 1xx:informational(request received, continuing process 2xx:success 3:redirection(further action needed to complete the request) 4: client error 5:server error
HTTP1.0: open new connection for every data item it retrieved, overhead in estab new connection to the same server over and over again. reuse connection over many req/.respon
request header: host(name of the server, used to implement virtual hosts),user-agent
response headers: server, content-length ,content type

NFS: Network File System. a client operate one file while its path changed? store info at client
locks -> lease: read lease may concurrently, write .. granted exclusively -> can eviction lease
RPC: client process->client stub(RPC library) -marshals-> clientOS -> s OS ->s stub->s pro
RPC fail: dup?xid. Too much xid history? client commit done and delete it. dup but original one still working? pending flag. Server crash? save its tables to disk rather than only in memory
Concurrent vs Parallel. thread shares same address space, thread context switch indep of OS.

Visualization: hardware change faster than soft. protability, isolation of failing components

Instruction set architecture(privileged and general), system calls, library calls.

DNS: host name – IP address ARP: IP -> MAC ,DHCP: MAC ->assign a unique IP

DHT(distributed hash table) hash table -> IP-> use RPC get file. Chord lookup: more efficient!

DNS: Root server, top-level domain(TLD) server, authoritative DNS server

Timing: Cristian’ s algoRTT=((T4-T1)-(T3-T2)) & Berkeley algo(average) & NTP minimum dispersion of RTT->choose server, minimum RTT -> choose time, Error: +/-1/2RTT

& Lamport clocks: a->b then C(a)<C(b),process i: C(a).i<C(b).j if C(a)<C(b) or C(a)=C(b) & i<j

& Vector clock: V(a)<V(z) then a->···->z, but C(a)<C(b) indicates nothing

CDN(usealias):original server->DNS TLD s->global DNS s->regional->nearby Akamai cluster

load balancer: selects a specific server(e.g. to maximize the cache hit rate)

moving content to the client is key: reduces latency, improve throughput, reliability

MTBF(Meantimebetweenfailures)MTTR(MeanTimeToRepair)Availability=(MTBF–MTTR)/MTBF

yield= queries completed/queries offered harvest=data available/complete data DQprinciple= Data per query * queries per second -> constant!!

reduce input->increase queries/s, adding nodes or software optimization changes constant

Standard internet routing: path chosen via Border Gateway Protocol

Why overlap? 1.for routing: better than best-effort service, reliability,2.to locate data:chord3.for security, encrypt the overlay, vpn

data flows along trees, why? improve reliability2.disseminate data in a scalable way3.avoid censorship

link stress: how often a packet...Relative delay penalty(aka stretch)delay overlay/underly net

HTTPS security, very high level basics of TLS. A security **certificate** is used as a means to provide the security level of a website to general visitors, Internet service providers (ISPs) and Web servers. **CA:** Certification authority. is an entity that issues digital certificates. one common use for certificate authorities is to sign certificates used in HTTPS, the secure browsing protocol for the WWW. a **cipher** is an algorithm for performing encryption or decryption

Network sockets API: open(), connect(), send(),recv()...How are domain names resolved to IP addresses? When you enter a URL into your Web browser, your DNS server uses its resources to resolve the name into the IP address for the appropriate Web server. ... Instead, you just connect through a domain name server, also called a DNS server or name server, which manages a massive database that maps domain names to IP addresses.EndtoEndProtocol:move from host2host ->process2process communication model. **TCP**provide abstraction of reliable in-order byte stream on top of IP protocol, **sliding windows**, **ACKs**. **Switching,forwarding**In general, forwarding refers to a device sending a datagram to the next device in the path to the destination, switching refers to moving a datagram from one interface to another within a device, and routing refers to the process a layer-3 device uses to decide on what to do with a layer-3 packet.**Idempotent**Operations that can safely be executed twice are called idempotent. **at - most-once delivery**means that for each message handed to the mechanism, that message is delivered zero or one times; in more casual terms it means that messages may be lost.**at-least-once delivery**means that for each message handed to the mechanism potentially multiple attempts are made at delivering it, such that at least one succeeds; again, in more casual terms this means that messages may be duplicated but not lost.**exactly-once delivery**means that for each message handed to the mechanism exactly one delivery is made to the recipient; the message can neither be lost nor duplicated. **A client stub** is responsible for conversion (marshalling) of parameters used in a function call and deconversion of results passed from the server after execution of the function. every RPC server and RPC client is linked with a copy of the **RPC runtime**. Runtime operations perform tasks such as controlling communications between clients and servers and finding servers for clients on request. An interface's client and server stubs exchange arguments through their local RPC runtimes. The client runtime transmits remote procedure calls to the server. **Clock synchronization** aims to coordinate otherwise independent clocks. **totally-ordered multicast**, i.e. all messages are delivered in the same order to each receiver. **balancer**It actually can balance the load (depends on the software of course). if a node crashes only the current connection dies (if at all) and the next request from the same client goes to a different node (vs. the client caches DNS and hits it again and again).**RR** maybe cheaper? **Replication** The challenge with replication is consistency. . data.**Partitioning** The challenge for partitioning is re-partitioning. Dynamically changing the number of partitions of the data can be an exceedingly expensive operation.