2.1 Principles

Creating a network app - when you create app layer, you write programs that: run on (different) end systems,communicate over network

Do not need to write software for network-core

- network-core devices do not run user applications -allows for rapid app development, propagation

Process communication

process: program running within a host, Port number sends to specific process

client process:process that initiates communication

server process:process that waits to be contacted

Sockets- process sends/receives messages to/from socket, socket like a mailbox

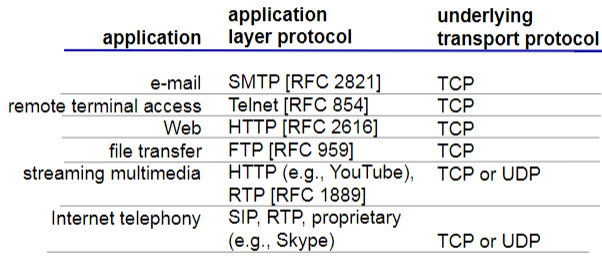
What does app-layer protocol define?

types of messages exchanged - request, response message syntax: what fields in messages & how fields are portrayed message semantics: meaning of information in fields rules for when and how processes send & respond to messages

Transport layer services-Needs to be reliable, fast/time sensitive,have throughput, secure

TCP (Transmission Control Protocol) - is reliable, has flow control, congestion control, does not provide timing, security, requires a setup between client and server Flow Control: sender won’t overwhelm receiver Congestion Control: throttle sender when network is overloaded

UDP (User Datagram Protocol) service- Unreliable, provides timing does not provide, flow control, congestion control, throughput guarantee, security



UDP exists because: no connection saves time, faster when there's no need to check reliability, and header is smaller than tcp

Securing TCP

\*SSL provides encrypted TCP connection, data integrity, end-point authentication. SSL is at app layer, Apps use SSL libraries, which “talk” w TCP

2.2 Web and HTTP

web page consists of objects:

* object can be any asset/file ...
* web page consists of base HTML-file which includes referenced objects
* each object is addressable by a URL

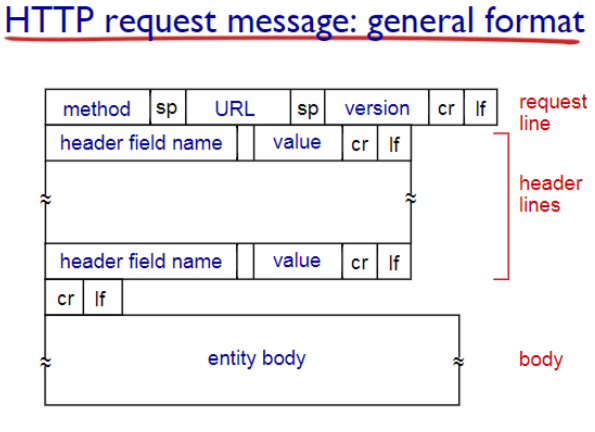
HTTP: hypertext transfer protocol - Web’s application layer protocol

client/server model (through HTTP Protocol):

client: browser that requests, receives and “displays” Web objects server: Web server sends objects in response HTTP is “stateless” - server maintains no information about past client requests

non-persistent HTTP -at most one object sent over TCP connection, connection then closed.

persistent HTTP -multiple objects can be sent over single TCP connection

HTTP request message: request line, header lines, carriage return: line feed at start indicate end 

HTTP response:

Status codes: 200 OK- request succeeded

301 Moved Permanently -requested object moved, 400 Bad Request request msg not understood

404 Not Found requested document not found on this server 505 HTTP Version Not Supported

Format: Status line w/ protocol response, header lines, data/requested file

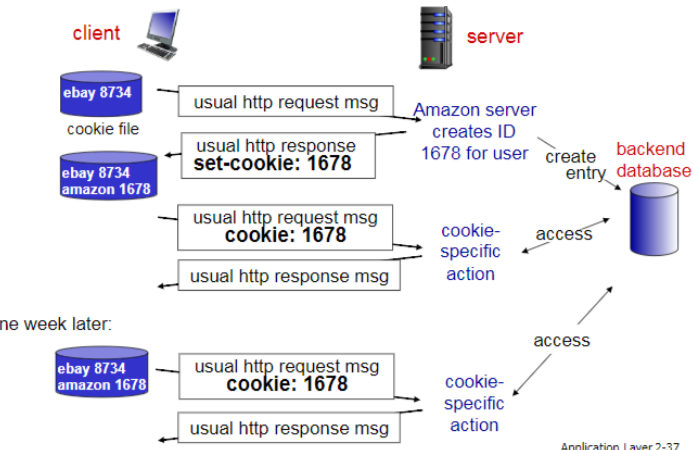
Response Methods:

HTTP/1.0: GET, POST | HTTP/1.1: GET, POST, HEAD, PUT, DELETE POST -input is uploaded to server in entity body URL method: uses

GET method, input is uploaded in URL

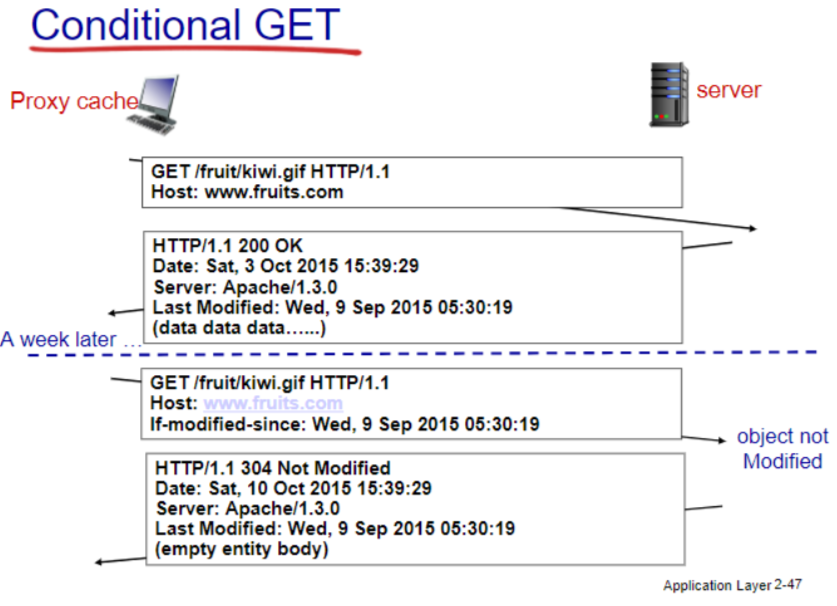
HEAD - asks server to leave requested object out of response PUT - uploads file in entity body to path specified in URL field |DELETE - deletes file specified in the URL field

Cookie: a method of keeping user/server state

Cookies used for: authorization, shopping carts, state (Web e-mail) , etc - Server stores cookie on database, client saves cookie file

Web cache - put data on closer server via cashing. Typically by isp or institution

why Web caching?: reduce response time, Reduce traffic on access link, enables “poor” providers to effectively deliver content, Cheap

Understand the example: local cache has smaller delay than vs widening access while being cheaper

Conditional GET - Goal: don’t send object if cache has up-to-date cached version Use the Conditional GET message to verify up to date. GET method: If-modified-since: <date>

2.3 email major components: user agents, mail servers.

simple mail transfer protocol: SMTP. SMTP uses TCP to reliably transfer, direct transfer: sending server to receiving server

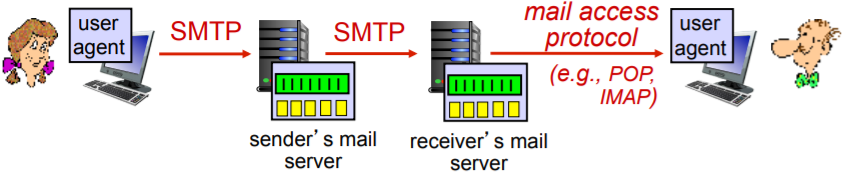
Mail message format: SMTP, RFC 822 & body: RFC 822 is standard for message header lines, e.x) To:, From, Subject: Different from SMTP MAIL FROM, RCPT TO: Body: the message in ASCII characters only

Mail access protocols:

POP3: Post Office Protocol authorization,

Download iMAP: Internet Mail Access Protocol more

Features,uses a server

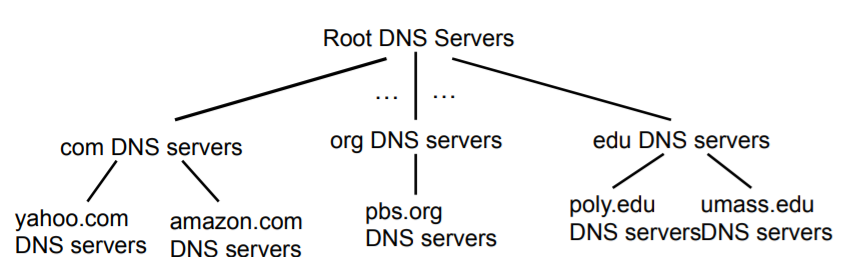
HTTP: online email system  
Differences: POP3 is stateless uses download & Delete, IMAP keeps all messages on server while keeping user state between sessions

For web, link between user and their server is HTTP

2.4 DNS DNS: Domain Name System

Services: Hostname to IP Address Translation (aka url), host aliasing, mail server aliasing, load distribution

Hierarchical structure:



Root: contacted by local if cannot resolve name. contacts authoritative name server if name mapping not known, gets mapping, returns mapping to local

TLD (top-level domain) servers: responsible for common domain like com, org, and for country lvl

authoritative DNS servers:organization’s own DNS server(s), providing authoritative hostname to IP mappings for organization’s named hosts

Local DNS: does not strictly belong to hierarchy

each ISP has one - when host makes DNS query, query is sent to its local DNS server, has local cache

- acts as proxy, forwards query into hierarchy

iterated query: contacted server replies with name of server to contact

recursive query: puts burden of name resolution on contacted name server

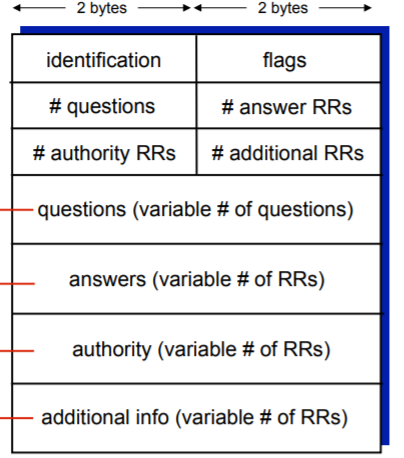
DNS caching: once (any) name server learns mapping, it caches/temp saves mapping

DNS records: distributed database storing resource records (RR): -(name, value, type, ttl)

type=A **name** is hostname § **value** is IP address

type=NS n**ame** is domain (e.g.,foo.com)

**value** is hostname of authoritative name

server for this domain

type=CNAME **name** is alias name for real name  **value** is canonical name

type=MX **value** is name of mailserver associated with **name**

Image is DNS message Format

DNS registrar inserts two RRs into .com TLD server: type A record for [www.\_url](about:blank), Type MX for \_url

Attacking DNS - DDoS Attacks bombard root or TLD server with traffic, redirect attacks intercept queries

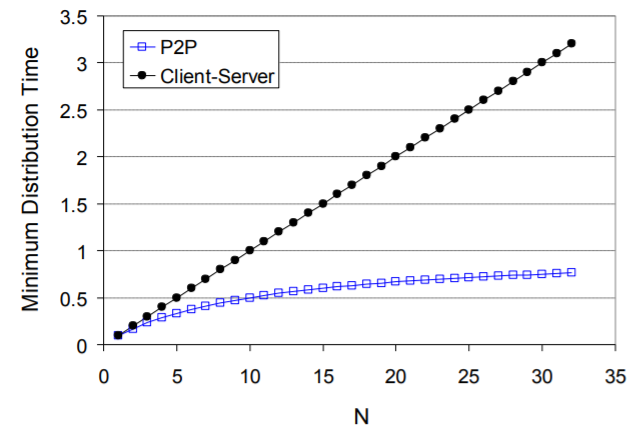
2.5 P2P Client-Server Architecture: Traditional network infrastructure with servers

server: always-on host | permanent IP address |

clients: communicate with server | intermittently connected | dynamic IP | do not communicate directly with other clients

P2P (Peer 2 Peer) architecture: no servers - peers host data & directly communicate | peers request & provide service from other peers| more scalable &self scalable – new peers bring new capacity| Drawback: - complex management bc peer is not a dedicated server

features of P2P - file distribution, streaming, VoIP

D = time, d = download capacity, u = upload capacity, F= file size, N= Peers, 

BitTorrent: P2P file distribution - gets chunks over time, registers with trackers to get peers

understand the mechanism

Tit-for-tat: While sending looks for top 4 peers, evaluates peer every 30s

2.6 video streaming and CDN

DASH(Dynamic, Adaptive Streaming over HTTP)

server: divides video file into multiple chunks, each chunk stored, encoded at different rates

manifest file: provides URLs for different chunks   
client: §periodically measures bandwidth •chooses maximum coding rate sustainable given current bandwidth

Content distribution Networks - store/serve multiple copies of videos at different places

ex) Netflix cx wants madman

CDN: stores copies of content at CDN nodes

When getting file from CDN - directed to nearby copy, retrieves content, may choose different copy if network path congested

Youtube - does not use DASH, uses DNS Redirect

Kankan - Hybrid CDN-P2P - CDN when P2P bad

2.7 socket programming

UDP: no “connection” between client and server

TCP: Client must contact server, doing so by creating tcp socket, specifying ip address, and port number of server process. Server creates new socket which allows talk with multiple clients.