CS2105 Live Class Lec2:Application layer

$_{/}|/|U_{C}h@NgRu!$

May 5, 2021

1 The application app

The program runs in application layer on endsystems (e.g. Web server software, browser software)
IMPORTANT: network-core devices do not run user app
allows for rapid application development, propagation

2 Client-server architecture

2.1 server

1. always on host(to serve the client)
2. permanent IP address(to ensure the client can find it)
3. data centers for scaling(needs a lot of machine to serve millions of clients simotaneously

2.2 client

1. communicate with server
2. may be intermittently connected
3. may have dynamic IP address
4. do not communicate directly with each other

Among the clients, they do not communicate with each other

3 P2P architecture

1. Each nodes behave both like a server and a client
2. no always-on server
3. arbitrary end systems directly communicate
4. peers request service from other peers, provide service in return to other peers
5. Self scalability: new peers bring new service capacity, as well as new service demands
6. peers are intermittently connected and change IP addresses
7. However, complex management

4 Process communication

1. Process: program running within a host(the application connections are between programs)
2. Within same host, two processes communicate using inter-process communication(IPC)
3. Processes in different hosts communicate by exchanging messages

4.1 client and servers

1. client process: process that initiates communication
2. server process: process that waits to be connected

4.2 client and servers relationship in P2P

Every side can be both client and server, and we usually call the side that initialize the connection client and the other side as server

5 Sockets

5.1 process sends/receives messages to/from its socket

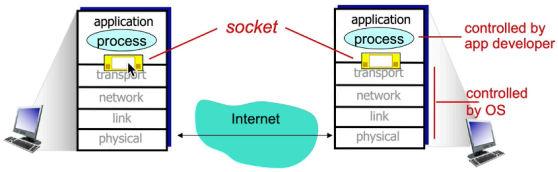
5.2 socket analogous to door

1. sending process shoves message out door
2. sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process

All in all, the socket is a interface for the application layer to access lower services

5.3 the abstraction

There is one file



As the application developer, he/she can control one side of the door, but do not get much control on the other side of the door(Type of services, and control parameter)
The socket is also called application programing interface(API)
Use API to write the application

6 Addressing processes

6.1 to receive messages

1. to receive messages, process must have identifier
2. host device has unique 32-bit IP address
3. IP address of host on which process runs is not suffice for identifying the process(same machine may run different applications, same application may run different processes

7 Identifier

1. Include both IP address and port number associated with process on hosts
2. Example port number

HTTP server: 80
mail server: 50

3. local host IP: 127.0. 0.1

8 Services an app need

8.1 data integrity

1. 100% reliable data transfer(with no loss or misorder)

vs

2. Loss tolerate

8.2 timing

require low delay (pornhub for e.g)

8.3 throughput

meaning: the amount of information should be large

8.4 security

9 two internet provided protocol to apply

9.1 TCP service

1. reliable
2. flow control:sender won’ t overwhelm receiver
3. congestion control:throttle sender when network is overloaded (not covered in cs2105)
4. no: timing, minimum throughput, guarantee, security(in some vision, there is)
5. connection-oriented

9.2 UDP service

1. unreliable
2. does not provide all thing that TCP does not provide and also does not provide reiability and connection setup

9.2.1 Why UDP is still used even though it is unreliable

There is always trade-off

1. the TCP connection takes time(audio)
2. some time the client really don’t know who to connect with, just broadcase instead(DHCP, DNS)
3. some services are loss tolerant(VoIP)

10 Some services and corresponding App-layer and trans layer

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

10.0.1 sender

The application is not the application protocol, the protocol defines the format and behavior and meaning of the application

11 Application protocol

1. **types of messages exchanged**(request, response)
2. **message syntax**,what fields in message how fields are delineated
3. **message semantics**(meaning of information in fields)
4. **rules** for when and how processes send respond to messages

12 Types of Application protocol

12.1 open protocols

1. defined in RFCs(Request for comments)
2. allows for interoperability (HTTP,SMTP)

12.2 proprietary protocols

E.g Skype: not open; can use reverse engineering to guess

13 HTTP protocol

web page consists of objects
object can be HTML file,JPEG image, Java Applet, audio file
web page consists of base HTML-gile. which includes several referenced objects
URL: address object (www.someschool.edu/someDept/pic.gif; host name+pathname)

14 HTTP overview

HTTP: hypertext transfer protocol

14.1 client/server model

14.1.1 client

browser that requests, receives(using HTTP protocol) and display Web objects

14.1.2 server

Web server sends (using HTTP protocol) objects in response to requests

14.1.3 function

Even if you do not use the same browser, under the same protocol, you are interact with each other

14.1.4 HTTP is stateless

Servers maintains no information about past client requests
The server does not keep track of the information of the client behavior
Reason: protocols that maintain ”state” are complex (past history must be maintained; if server/-client crashes, their views of ”state” may be inconsistent, must be reconciled
In reality, extra mechanism beside HTTP is applied to maintain the state of client, which is called cookie

15 HTTP connection

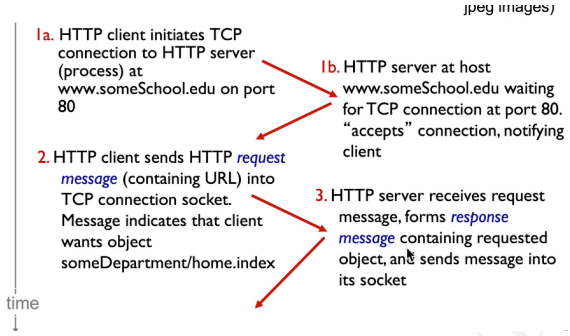
15.1 non-persistent HTTP

1. at most one **object** sent over TCP connection, connection then closed
2. downloading multiple objects required multiple connection

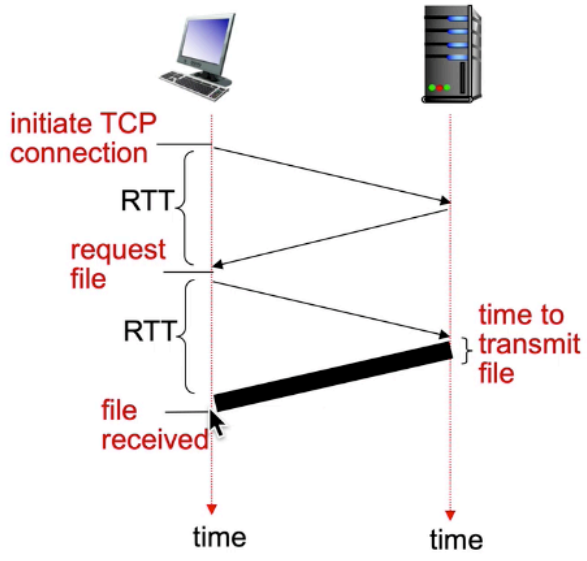
15.2 persistent HTTP

multiple objects can be sent over single TCP connection between client, server

16 The step flow of the HTTP



17 The time cost for the HTTP connection



17.1 RTT(definition): time for a small packer to travel from client to server and back

17.2 HTTP response time

1. one RTT to initiate TCP connection
2. one RTT for HTTP request and first few bytes of HTTP response to return
3. file transmission time
4. non-persistent HTTP resoonse time =2RTT+file transmission time(The point is that during the RTT, the client should wait for the response)

If the file is large, the HTTP the divide the packet into chunks and sent one by one, each takes at least two RTT

18 persistent HTTP response time(the default model of the HTTP)

1. server leaves connection open after sending response
2. subsequent HTTP messages between same client/server sent over open connection
3. client sends requests as soon as it encouters a referenced object
4. as little as one RTT for all referenced objects(still one RRT at the begining)

But the server side usually decide when to close the connection(there is a timer)
IMPORTANT: The HTTP persistent connections do not use separate keep-alive messages, they just allow multiple requests to use a single connection.(i.e. persistent connection just means multiple request-response through the same keep-alive connection in sequence but not simultaneously

18.1 parallel connection

It's introduced as a way to improve the non-persistent HTTP, but there is no statement found that the persistent HTTP cannot use parallel, the fact is that out computer make multiple HTTP simultaneously, Even without any ordering, it can't be optimal without large enough available bandwidth. As a solution, browsers open several connections to each domain, sending parallel requests.
reference:https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x

18.2 parallel and pipelining

parallel means multiple connections between the server and client while pipelining(a concepts opposite to stop-and-wait manner) means the sender is allowed to send multiple packets without waiting for acknowledgments

18.2.1 parallel

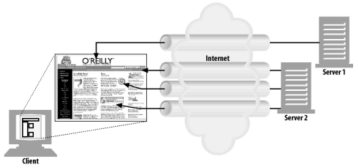
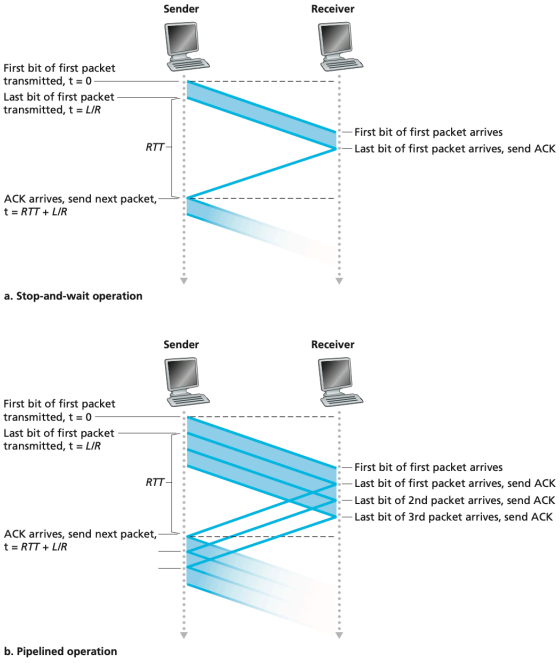


Figure 4-11. Each component of a page involves a separate HTTP transaction

18.2.2 pipelining



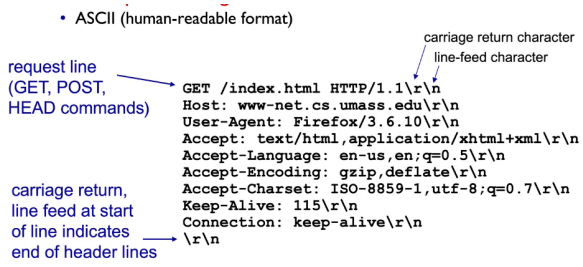
18.2.3 Difference between persistent, non-persistent, pipelining and parallel

If A wanna send 10 pkts with size1 to B

1. non-persistent HTTP (RRT+RRT+$\frac{size1}{transmissionrate}$)*10
2. persistent HTTP(stop&wait) RTT+(RRT+$\frac{size1}{transmissionrate}$)*10
3. persistent HTTP(pipeline): RRT+RRT+($\frac{size1}{transmissionrate}$)*10
4. non-persistent HTTP(x parallel):(RRT+RRT+$\frac{size1}{transmissionrate}$)*10/x

19 two types of HTTP message

19.1 request



19.1.1 In the first line:

1. the first is the order(e.g. GET,POST,HEAD)
2. the second is the URL
3. the third is the HTTP vision(1.1 means defautly persistent)

IMPORTANT: the 1.1/1.0 are HTTP visions not just the persistent mode

The 1.1 is persistent defautly while 1.0 is in-persistent

1.0 can also be set persistent by setting Connection to anything other than close(usually retry-after)

reference:https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x

19.1.2 the middle part: header part

Each lines in header part are called header lines

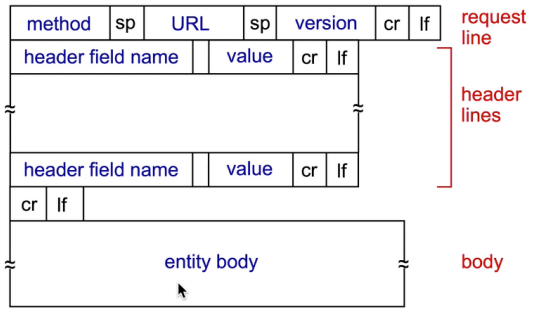
In the header field, each part gives both the name(key) and corresponding content(value)

e.g the "Connection:keep-alive\ r \n also means persistent The part "Keep-Alive

e.g the User-Agent tells what kind of browser is used

e.g the Keep-Alive: 115\ r \n ẗells the time of connection for persistent The "\ r \n" in the last line tells the end of the header

19.2 general form of HTTP request



The body part is usually empty for GET method
The POST method and URL method need body part (need to submit keywords or parameters)

19.2.1 POST

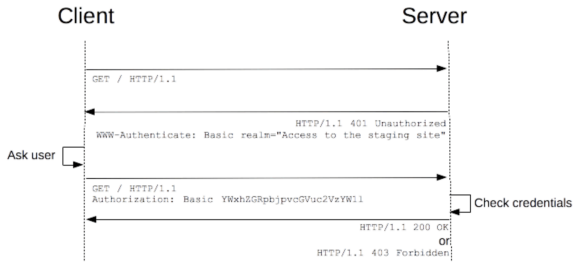
form input
input is uploaded to server in entity body

19.2.2 URL method

1. uses GET method
2. input is uploaded in URL field of request line

19.3 Additional: control and authentication

HTTP provides a general framework for access control and authentication.



reference: https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

19.4 method types

19.4.1 HTTP/1.0

1. GET
2. POST
3. HEAD(ask server to leave requested object out of response) if the client call HEAD to a server, the server will response the same message(the only change is the content part will be left empty) the purpose of HEAD is usually just for debuging(so content not needed) to make sure that what the server get is right.

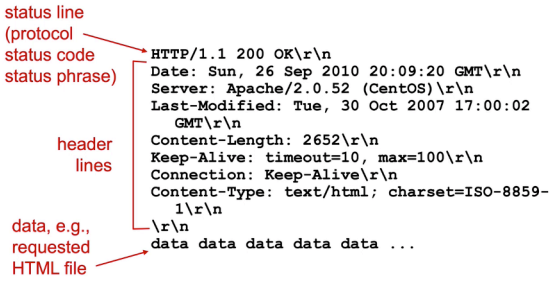
19.4.2 HTTP/1.1

1.GET, POST, HEAD
2. PUT(upload file in entity body to path specified in URL field
3. DELETE(deletes files specified in URL field)

19.4.3 other method

https://www.w3schools.com/tags/ref_httpmethods.asp

19.5 response



1. The first line: the status line(protocol status code statuse phrase)
2. header: similar to the request head
3. body part: data e.g. requested HTML file

19.6 status code

200: OK

301: Moved permanently

308 Permernant redirection

400: Bad request

404: Not Found

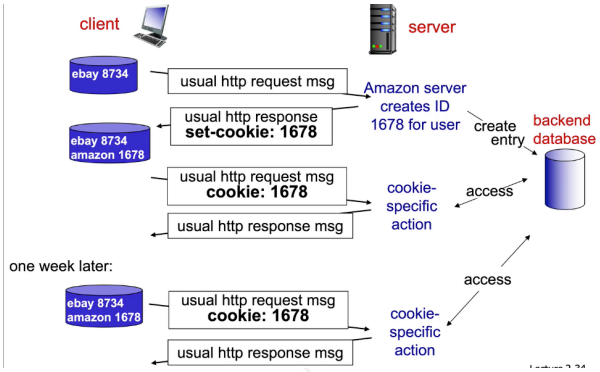
505: HTTP Version Not Supported

The 301 and 308 can be used for permanent redirection, which tells the client that the resources have been moved to another given URL website

20 User-server:cookie

20.1 Four components:

1. cookie header line of HTTP **response** message
2. cookie header line in next HTTP **request** message
3. cookie file kept on user's host , managaged by user's browser
4. back-end database at Website



1. Suppose there is one people visited website: ebay and his/her computer already have a cookie called ebay; suppose she/he first time visit Amazon web;

2. the amazon notice that it's a new user and keep track of the user, and create a new ID(1678) (though the amazon does not really the identity of the user) and corresponding entry in the database to keep track of the new user(cookie mechanism start from when the server create the entry)

3. In the response message, it will put one header line, the key "set-cookie", value is the id the server create for the user(i.e. 1678)

4. Later, when the user visit the website again, the website can recognize it with the ID(even

though the wesite does not know the identity of the user); but it can still retrieve the data information of the user before
The actual state is stored in the database; the client site only knows the ID

20.2 What cookie can be used for

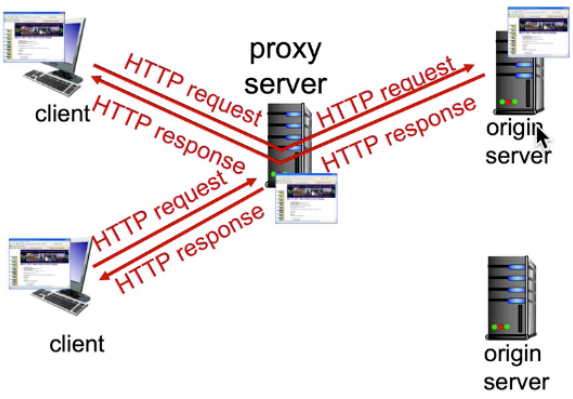
1. authorization
2. shopping carts
3. recommendation
4. user session state (Web email)

20.3 The way to keep the state

1. protocol endpoints: maintain state at sender/receiver over multiple transaction
2. cookies: http message carry state

20.4 Web caches(proxy server)

goal: satisfy client request without involving origin server
1. user sets browser: web accesses via cache
2. browser sends all HTTP requests to cache:
a. object in cache cache returns object
b. else cache requests object from origin server, then returns object client

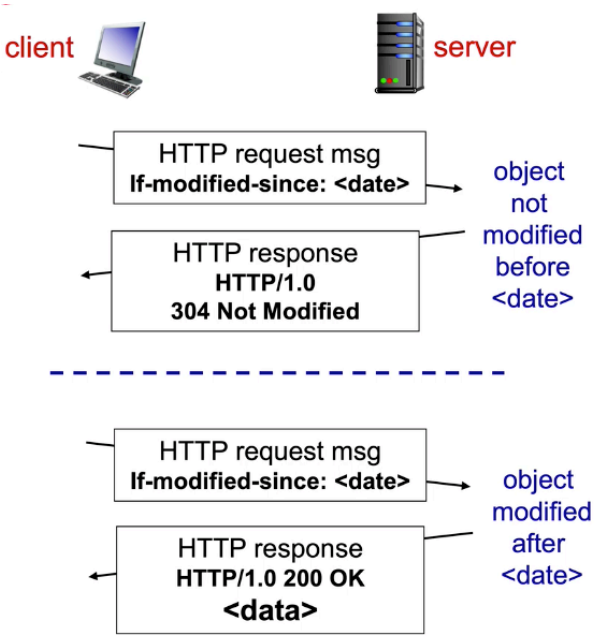


When a client make certain request, the request is redirect to a proxy server, if the request information is all stored in proxy server, the it will response directly; if the request information is not all in the proxy server, it will promote the request to the original server and get the response and send to the client(it will cache the request and response)

20.5 Conditional GET

20.5.1 To solve the less updated problem of the proxy server

1. The proxy server will send requests to the original server like a client and to check whether the object is modifies or not in a subsequent time
2. If the response contains no object, it means that the object in proxy server is up-dated; If the the object in proxy server is not updated, the original server will send back response with the updated data



21 DNS(domain name system)

21.1 Internet has a unique identify with two formation

1. IP address(32 bit) used for addressing datagrams, e.g. 137.132.80.57
2. **Hostname** used by human, e.g. www.comp.nus.edu.sg

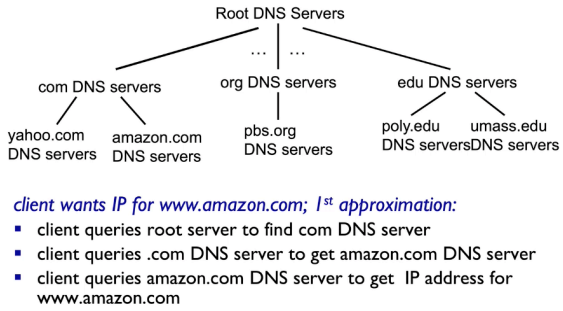
21.2 Domain name system

1. distributed database implemented in hiearchy of many name servers
2. application-layer protocol: hosts, name servers communicate resolve names(address/name translation)
3. A server-to-server query or response - at least with UDP, where both source and destination port are 53; with TCP, the requesting server will use a port above 1023.

a. core Internet function, implemented as application layer protocol
b. complexity at network's "edge" (it's not inside the network core)

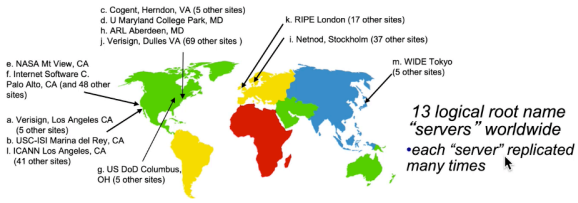
21.3 Local DNS name server

when host makes DNS query, query is sent to its local DNS server
has local cache of recent name-to-address translation pairs(but not necessarily updated)
acts as proxy, forward query into hierachy(which means where may be proxy of proxies)
IMPORTANT: each internet service provider(e.g. Singtel, starhub) has at least one DNS name server; (all these servers may be connected in a hierachy structure)
However, these ISP's local DNS service may not belongs to the worlds' hierachy



If the local DNS server does not have the mapping, it will go through the searching tree structure and tries to find the translation in the global DNS server hierachy

22 DNS entities



22.1 DNS root name server

Very top

1. contacted by local name server that can not resolve name

2. root name server: over 400 servers all over the world; provide IP address of TLD server ;

There are many data center to provide the top level service the second level

22.2 TLD

Top-level domain(TLD): responsible for com, org, net, edu, aero, jobs, museum

e.g network solutions maintains servers for ".com" TLD

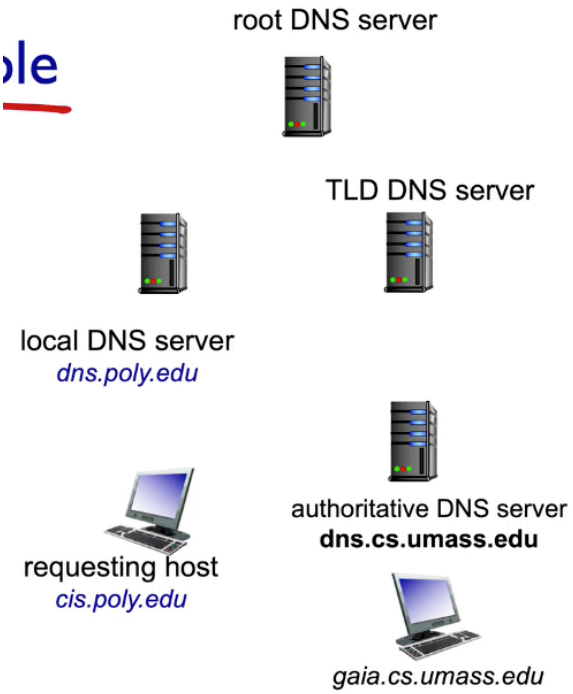
Educause for .edu TLD

22.3 authoritative DNS servers:

1. organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts

2. can be maintained by organization or service provider(e.g www.baidu.com; www.nus.edu.sg)

23 DNS resolution example



1. host at cis.poly.edu wants IP address for gaia.cs.umass.edu, the browser make the query to the local DNS name server(dns.poly.edu)
2. the local DNS server cannot find it in the cached entries, and hence make a query to root DNS server(like a client)
3. the root DNS server will give local DNS server the IP address to the TLD DNS server
4. the local DNS server make a query to the given TLD DNS server
5. the TLD DNS server gives the local DNS server the IP address to the authoritative DNS server
6. the local DNS server make a query to the authoritative DNS server which is dns.cs.umass.edu
7. The local DNS will get the translation and cached it
8. the local DNS server will send the updated translation to the reauesting host

23.1 Two types of query

23.1.1 iterated query

Like the query 2 and query 4 in the example
contacted server replied with name of server if they know it, but they don't give the final answer

like" I don't know the answer, but I know who know

23.1.2 recursive query

puts burden of name resolution on contacted name server
it will eventually gives the final answer
In reality, this does not happen, because it will put much heavy work load at upper levels, so it's needed to seperate the stress

23.2 Once name server learns mapping, it caches mapping

cache entries timeout(disappear) after some time(TTL)
TLD servers typically cached in local name servers(thus root name server not often visited)

23.3 cached entries may be out-of-date(best effort name-to-address translations)

if name host changes IP address, may not be known Internet-wide until all TTLs expire(a suitable TTL value is essential) (that's why if we just remeber the website, it may not work after certain time)
There is a update/notify mechanisms proposed IETF standard(RFC 2136)
The service provider also monitor the usage amount and see whether should update

23.4 dig - DNS lookup utility

try "man dig" in terminal to check the function

```
(base) r-136-105-25-172:~ mcr$ dig -t A www.souhu.com

; <<>> DiG 9.10.6 <<>> -t A www.souhu.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50477
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;www.souhu.com.                 IN      A

;; ANSWER SECTION:
www.souhu.com.          3600    IN      CNAME   cndm.com.
cndm.com.               3600    IN      A       47.74.55.53

;; AUTHORITY SECTION:
cndm.com.               172800  IN      NS      ns2.dnbiz.com.
cndm.com.               172800  IN      NS      ns1.dnbiz.com.

;; Query time: 716 msec
;; SERVER: 192.168.140.2#53(192.168.140.2)
;; WHEN: Mon Mar 01 14:09:59 +08 2021
;; MSG SIZE  rcvd: 119
```

```
(base) r-136-105-25-172:~ mcr$ dig -t A www.pornhub.com

; <<>> DiG 9.10.6 <<>> -t A www.pornhub.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21986
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 8, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;www.pornhub.com.               IN      A

;; ANSWER SECTION:
www.pornhub.com.        3600    IN      CNAME   pornhub.com.
pornhub.com.            3600    IN      A       66.254.114.41

;; AUTHORITY SECTION:
pornhub.com.            80927   IN      NS      dns1.p03.nsone.net.
pornhub.com.            80927   IN      NS      dns2.p03.nsone.net.
pornhub.com.            80927   IN      NS      dns3.p03.nsone.net.
pornhub.com.            80927   IN      NS      dns4.p03.nsone.net.
pornhub.com.            80927   IN      NS      sdns3.ultradns.biz.
pornhub.com.            80927   IN      NS      sdns3.ultradns.com.
pornhub.com.            80927   IN      NS      sdns3.ultradns.net.
pornhub.com.            80927   IN      NS      sdns3.ultradns.org.

;; ADDITIONAL SECTION:
sdns3.ultradns.biz.     74864   IN      A       156.154.142.3
sdns3.ultradns.org.     74864   IN      A       156.154.143.3

;; Query time: 39 msec
;; SERVER: 192.168.140.2#53(192.168.140.2)
;; WHEN: Mon Mar 01 14:09:13 +08 2021
;; MSG SIZE  rcvd: 317
```

Here can see,
When execute ”dig -t a www.pornhub.com” comparing with ”dig -t a www.souhu.com”
It’s obvious that the query time for pornhub is significantly less, which indicates that no SOCer used souhu recently but some friend used pornhub recently

23.5 A good example of DNS look up

```
dig  a3.nstld.com +trace
```

issued from within NUS.

```
; <<>> DiG 9.2.2 <<>> a3.nstld.com +trace
;; global options:   printcmd
.                       162410  IN      NS      M.ROOT-SERVERS.NET.
.                       162410  IN      NS      A.ROOT-SERVERS.NET.
.                       162410  IN      NS      B.ROOT-SERVERS.NET.
.                       162410  IN      NS      C.ROOT-SERVERS.NET.
;; Received 436 bytes from 137.132.90.2#53(137.132.90.2) in 27 ms

com.                    172800  IN      NS      C.GTLD-SERVERS.NET.
com.                    172800  IN      NS      D.GTLD-SERVERS.NET.
com.                    172800  IN      NS      E.GTLD-SERVERS.NET.
com.                    172800  IN      NS      F.GTLD-SERVERS.NET.
;; Received 490 bytes from 202.12.27.33#53(M.ROOT-SERVERS.NET) in 296 ms

a3.nstld.com.           172800  IN      A       192.5.6.32
nstld.com.              172800  IN      NS      a2.nstld.com.
nstld.com.              172800  IN      NS      c2.nstld.com.
nstld.com.              172800  IN      NS      d2.nstld.com.
;; Received 277 bytes from 192.26.92.30#53(C.GTLD-SERVERS.NET) in 246 ms
```

1. a root DNS server: 202.12.27.33
2. a top-level domain DNS server: 192.26.92.30
3. a local DNS server: 137.132.90.2
4. host a3.nstld.com: 192.5.6.32
5. Here, there is only one authoritative DNS server for a3.nstld.com; but there are three author-

itative DNS server for nstld.com

23.6 subnet mask

23.6.1 function

A subnet mask is a number that defines a range of IP addresses available within a network. A subnet mask hides (or masks) the network part of a system's IP address and leaves only the host part as the machine identifier.
In simple, the subnet just tells how many bits are public IP

23.6.2 types

Default subnet masks are used with Class A, Class B, and Class C IP addresses, as follows:
Class A: 255.0.0.0
Class B: 255.255.0.0
Class C: 255.255.255.0

23.6.3 Legal

To judge whether a subnet mask is legal or not
Transfer each parts seperated by '.' into binary and join parts together, the result should be in the form of 111.1100..0

23.7 Redirect

A redirect is a way to send both users and search engines to a different URL from the one they originally requested. The three most commonly used redirects are 301, 302, and Meta Refresh.
IMPORTANT:The transportation protocol will not be changed, just change the URL
reference: https://moz.com/learn/seo/redirection