# Domain Name System (DNS)

Smith College, CSC 249
Feb 6, 2017



## TODAY: Domain Name System

- The directory system for the Internet
  - Used by other application layer protocols
  - … via socket programming
- Maps a hostname to an IP address
  - Host names use natural, human, language
    - URL such as www.google.com
  - IP addresses are numerical locators used by computers (more detail later)

## Application Layer Task

- You want your host (laptop, phone…) to
  - Send an email message
  - Retrieve a web page
- How do you find the equivalent of the actual, physical 'street address' of the destination host (the IP address)?
- DNS – nested, hierarchical loop-up system

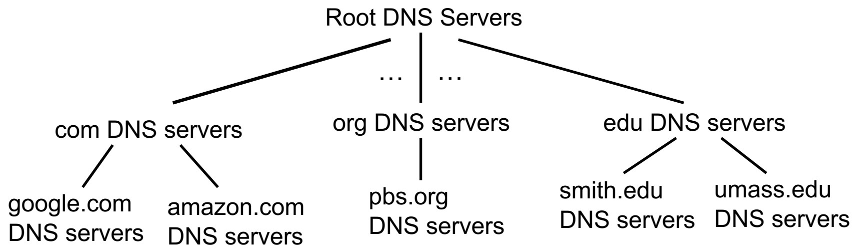
3

## Domain Name Servers

- Root Name Server
  - Top Level Domain Server
    - Authoritative Server
      - Local Name Server
        - Your computer looking for an IP address

4

## DNS: a distributed, hierarchical database




*a host, or client, wants the IP address for www.google.com*

1) Client (local server) queries root server to find the .com DNS server
2) Client queries .com DNS server (TLD) for google.com DNS server
3) Client queries google.com DNS server (authoritative) to get the IP address for www.google.com

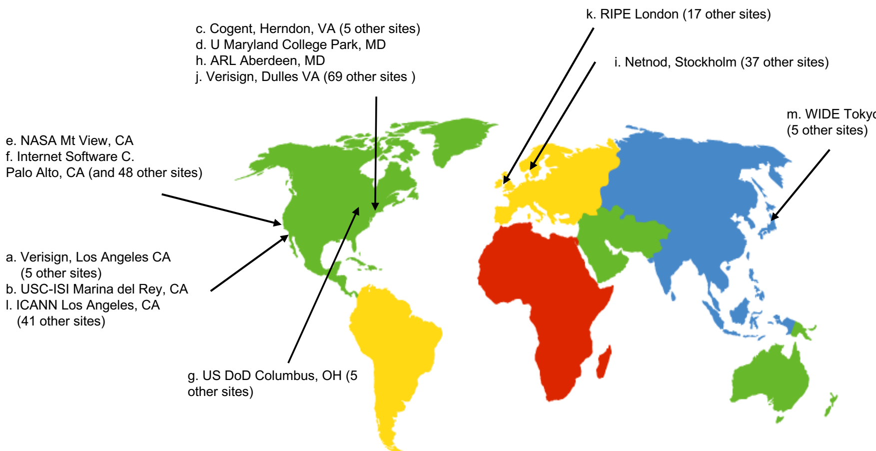


## DNS: root name servers

- The root name server is contacted by local name server in order to start finding the IP address
- root name server:
  - contacts TLD name server if name mapping not known
  - gets mapping and returns mapping to local name server (which will continue seeking)

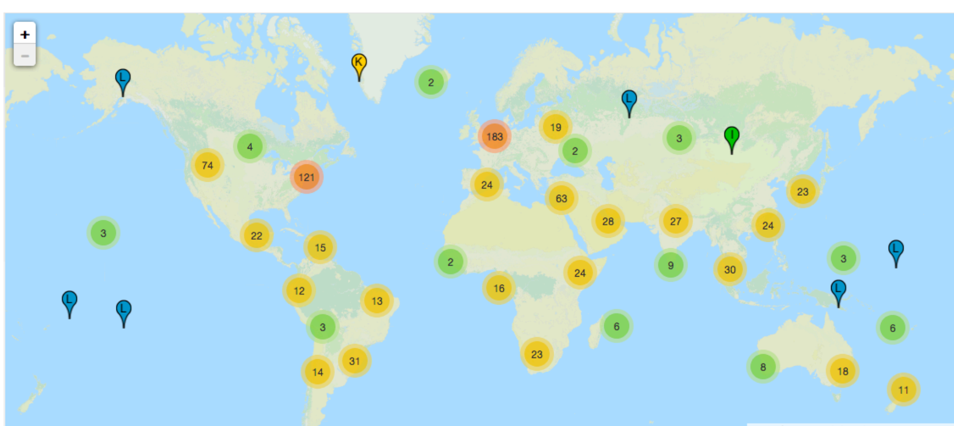
# DNS: root name servers

There are many logical root name "servers" worldwide, each "server" replicated many times (not shown: Russia, India, Australia, S. Africa, Brazil...)




# http://www.root-servers.org/

Interactive map:




8

## TLD & Authoritative Servers

*top-level domain (TLD) servers:*

- responsible for maintaining records mapping IP addresses for the DNS servers for .com, .org, .net, edu, and all top-level country domains, e.g.: uk, fr, ca, jp
- For example
  - Verisign Global Network Services maintains servers for .com TLD
  - Educause for .edu TLD

*authoritative DNS servers:*

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider



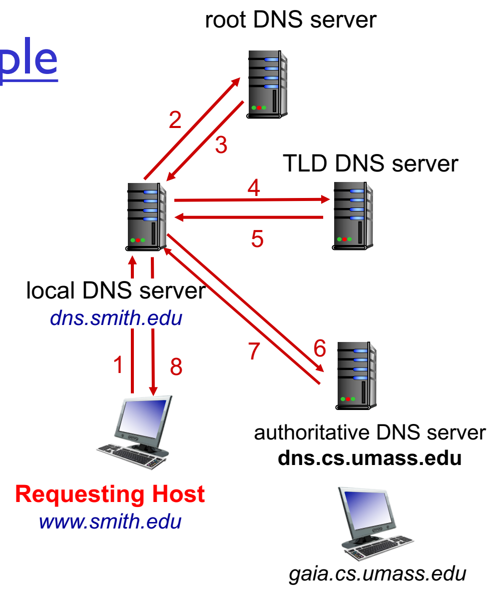
## Local DNS name server

- (does not strictly belong to hierarchy)
- Each ISP (residential ISP, company, university) has its own local DNS server
  - also called "default name server"
- When a host makes a DNS query, the query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date)
  - acts as proxy, forwards query into hierarchy
  - When you connect to network, your host is given the IP address of the local DNS server

# DNS name resolution example

- host at www.smith.edu wants IP address for gaia.cs.umass.edu

*iterated query:*
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"





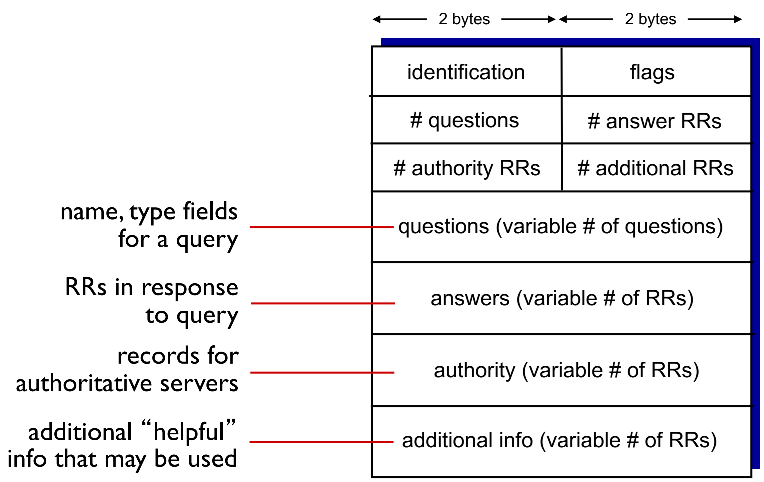

# DNS protocol, messages

- *query* and *reply* messages, both with same *message format*

Message header
- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative

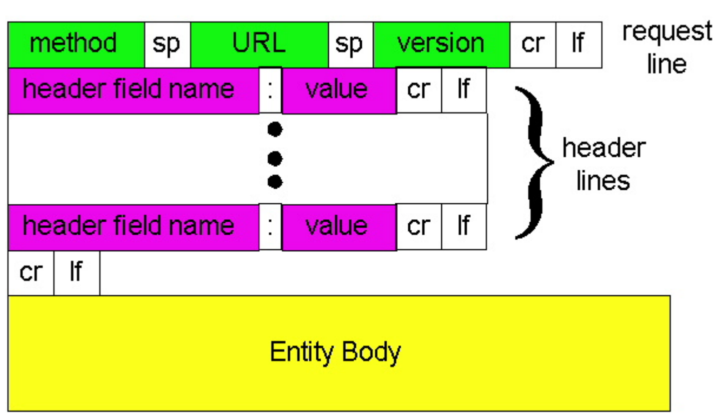
| 2 bytes | 2 bytes |
|---|---|
| identification | flags |
| # questions | # answer RRs |
| # authority RRs | # additional RRs |
| questions (variable # of questions) | |
| answers (variable # of RRs) | |
| authority (variable # of RRs) | |
| additional info (variable # of RRs) | |

# DNS protocol, messages





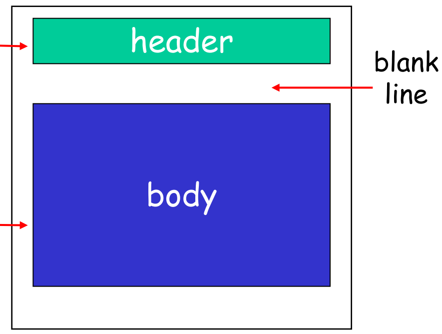

# HTTP request message: format



# Mail message format

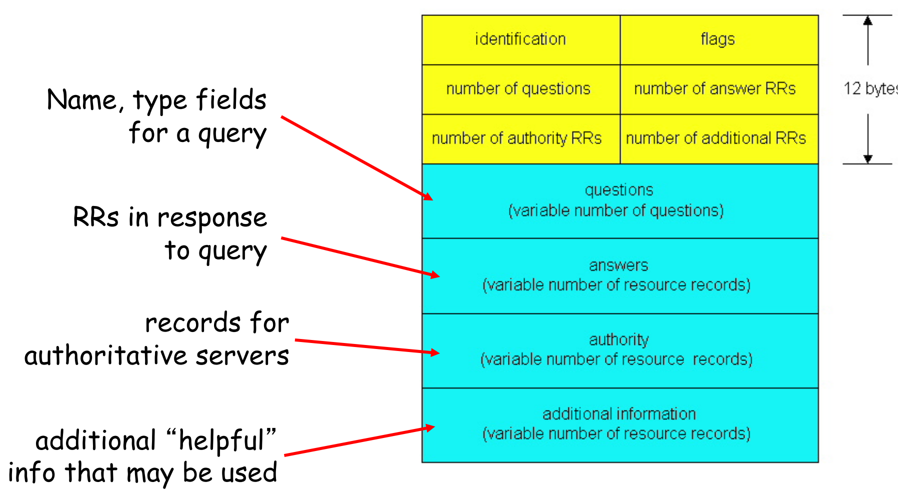
- Message header lines, e.g.,
  - To:
  - From:
  - Subject:

  *different* *from SMTP commands*!
- body
  - the "message", ASCII characters only




15

# DNS protocol, messages




16

```
[ford352-r10578:~ jcardell$ dig root-servers.org

; <<>> DiG 9.8.3-P1 <<>> root-servers.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63593
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; QUESTION SECTION:
;root-servers.org. IN A

;; ANSWER SECTION:
root-servers.org. 120 IN A 193.0.6.136

;; AUTHORITY SECTION:
root-servers.org. 3388 IN NS sns-pb.isc.org.
root-servers.org. 3388 IN NS sec2.authdns.ripe.net.
root-servers.org. 3388 IN NS ns.maxgigapop.net.

;; ADDITIONAL SECTION:
sec2.authdns.ripe.net. 11896 IN A 193.0.9.4
sec2.authdns.ripe.net. 11896 IN AAAA 2001:67c:e0::4
ns.maxgigapop.net. 87 IN A 206.196.176.2
ns.maxgigapop.net. 87 IN AAAA 2001:468:c00:6:225:90ff:fe72:119c
sns-pb.isc.org. 5939 IN A 192.5.4.1
sns-pb.isc.org. 5939 IN AAAA 2001:500:2e::1

;; Query time: 15 msec
;; SERVER: 131.229.64.2#53(131.229.64.2)
;; WHEN: Mon Feb  5 13:41:52 2018
;; MSG SIZE  rcvd: 270
```

17

"IN" is a rarely used 'class' field, and indicates "Internet"
#s indicate TTL

```
[ford352-r10578:~ jcardell$ dig root-servers.org

; <<>> DiG 9.8.3-P1 <<>> root-servers.org

;; QUESTION SECTION:
;root-servers.org. IN A

;; ANSWER SECTION:
root-servers.org. 120 IN A 193.0.6.136

;; AUTHORITY SECTION:
root-servers.org. 3388 IN NS sns-pb.isc.org.
root-servers.org. 3388 IN NS sec2.authdns.ripe.net.
root-servers.org. 3388 IN NS ns.maxgigapop.net.
```

18

"IN" is a rarely used 'class' field, and indicates "Internet" ford352-r10578:~ jcardell$ **dig smith.edu**

```
; <<>> DiG 9.8.3-P1 <<>> smith.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31681
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;smith.edu. IN A

;; ANSWER SECTION:
smith.edu. 21600 IN A 131.229.64.19

;; AUTHORITY SECTION:
smith.edu. 21600 IN NS ns1.smith.edu.
smith.edu. 21600 IN NS ns1.umass.edu.
smith.edu. 21600 IN NS babel.smith.edu.
smith.edu. 21600 IN NS ns2.umass.edu.
smith.edu. 21600 IN NS ns3.umass.edu.

;; ADDITIONAL SECTION:
ns1.smith.edu. 21600 IN A 198.101.218.79
ns1.umass.edu. 6636 IN A 128.119.10.27
ns2.umass.edu. 6636 IN A 128.119.10.28
ns3.umass.edu. 6636 IN A 128.103.38.68
babel.smith.edu. 21600 IN A 131.229.64.2

;; Query time: 0 msec
;; SERVER: 131.229.64.2#53(131.229.64.2)
;; WHEN: Mon Feb  5 13:44:14 2018
;; MSG SIZE  rcvd: 221
```

19

ford352-r10578:~ jcardell$ **dig mail.smith.edu**

```
; <<>> DiG 9.8.3-P1 <<>> mail.smith.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4657
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 8

;; QUESTION SECTION:
;mail.smith.edu. IN A

;; ANSWER SECTION:
mail.smith.edu. 21600 IN CNAME ghs.google.com.
ghs.google.com. 204 IN A 172.217.9.243

;; AUTHORITY SECTION:
google.com. 1415 IN NS ns2.google.com.
google.com. 1415 IN NS ns3.google.com.
google.com. 1415 IN NS ns4.google.com.
google.com. 1415 IN NS ns1.google.com.

;; ADDITIONAL SECTION:
ns2.google.com. 17146 IN A 216.239.34.10
ns2.google.com. 285318 IN AAAA 2001:4860:4802:34::a
ns1.google.com. 106901 IN A 216.239.32.10
ns1.google.com. 285318 IN AAAA 2001:4860:4802:32::a
ns3.google.com. 17146 IN A 216.239.36.10
ns3.google.com. 285318 IN AAAA 2001:4860:4802:36::a
ns4.google.com. 17146 IN A 216.239.38.10
ns4.google.com. 112513 IN AAAA 2001:4860:4802:38::a

;; Query time: 0 msec
;; SERVER: 131.229.64.2#53(131.229.64.2)
;; WHEN: Mon Feb  5 13:45:28 2018
;; MSG SIZE  rcvd: 324
```

20

## DNS record format

The distributed database stores resource records (RR)

RR format: `(name, value, type, ttl)`

- Type=A
  - `name` is hostname
  - `value` is IP address
- Type=NS
  - `name` is domain (e.g. smith.edu)
  - `value` is hostname of authoritative name server for this domain
- Type=CNAME
  - `name` is alias name for some "canonical" (the real) name
    `www.ibm.com` is really
    `servereast.backup2.ibm.com`
  - `value` is canonical name
- Type=MX (mail server)
  - `value` is name of mailserver associated with `name`

21

## DNS records

DNS: distributed db storing resource records (RR)

RR format: `(name, value, type, ttl)`

`(hostname, IP address, A, ttl)`

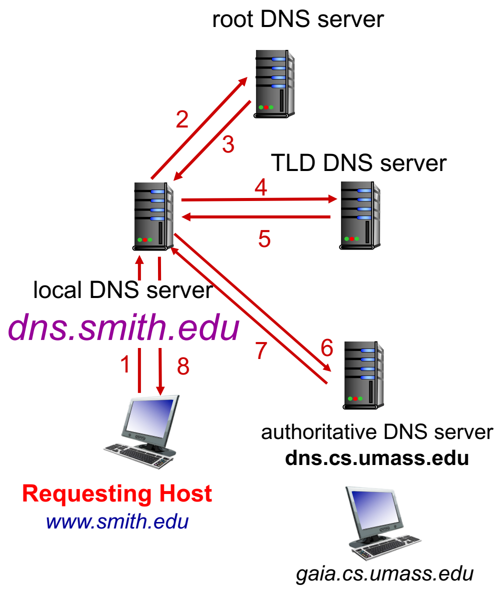
`(domain, hostname-DNS-author-server, NS, ttl)`

`(alias hostname, canonical name, CNAME, ttl)`

`(alias hostname, mail server cname, MX, ttl)`

22

# DNS Records




+ Type NS record with hostname of auth. server for the requested domain name

+ *and* Type A record for IP address of auth. server

Type A record for IP address of 'hostname'

# * Investigate the DNS process *

DNS protocol : *query* and *reply* messages, both with same *message format*

Message header

- identification: 16 bit # for query, reply to query uses same #
- flags
- Number of records in the message itself

| | |
|---|---|
| identification | flags |
| number of questions | number of answer RRs |
| number of authority RRs | number of additional RRs |
| questions (variable number of questions) | |
| answers (variable number of resource records) | |
| authority (variable number of resource records) | |
| additional information (variable number of resource records) | |

12 bytes

- Try:
  - >> dig <...>
  - >> nslookup <...>

# nslookup at terminal prompt

**ford352-r10578:~ jcardell$ nslookup mail.smith.edu**
Server: 131.229.64.2
Address: 131.229.64.2#53

mail.smith.edu canonical name = ghs.google.com.
Name: ghs.google.com
Address: 172.217.9.243

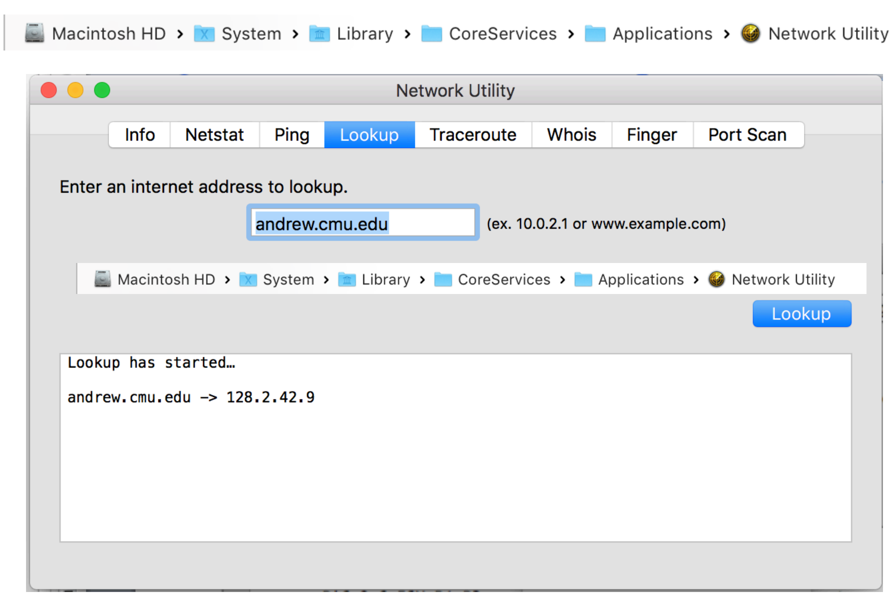
**********************************************************************

**ford352-r10578:~ jcardell$ nslookup science.smith.edu**
Server: 131.229.64.2
Address: 131.229.64.2#53

Name: science.smith.edu
Address: 131.229.64.139

25

# nslookup with Mac OS




26

## Summary of Application Design Elements

- Message format
  - ASCII? Binary?
  - How handle (send) multiple objects?
- Number of connections
  - Persistent? Parallel connections?
- State information? Stateless?
- TCP or UDP used (Transport Layer)?
- Push or pull protocol?
- How to find the server? client? peer?
- Handshaking in the protocol?
- Centralized? Decentralized? (peer-to-peer)

27

## First View of Sockets

28

## Sockets – analogous to file I/O

- Three steps in file I/O
  1) open the file – associate a file on your disk with a variable in your program
  2) read and write – set of operations to manipulate the file contents – the file associated with your file variable
  3) close the file – ensure changes actually written to the disk, ensure other programs can access and use the file, dissociate the file and the variable

29

## Sockets – analogous to file I/O

- Python File I/O Syntax
  - <filevar> = open( <filename>, <mode>)
    - open() returns a file object
    - mode = 'r', 'w', 'a'

30

## Sockets - file I/O (EM)

```
# Example of Python file I/O
outFile = open("myFile.txt", 'w')
outFile.write("Hello CSC111!\n")
outFile.write("Files are fun!!\n")
outFile.close()

infile = open('myFile.txt', 'r')
text = infile.read()
infile.close()

print text
```

31

## Sockets - file I/O (DT)

```
# Example of Python file I/O
# write some variables to file
# your unique input:
name = "Smith College"
address = "Elm st., Northampton, MA 01063"

# Python file I/O commands
file = open( "college.txt", "w" )
file.write( "%s\n" % name )
file.write( "%s\n" % address )
file.close()
```

32

## Sockets – file I/O (DT)

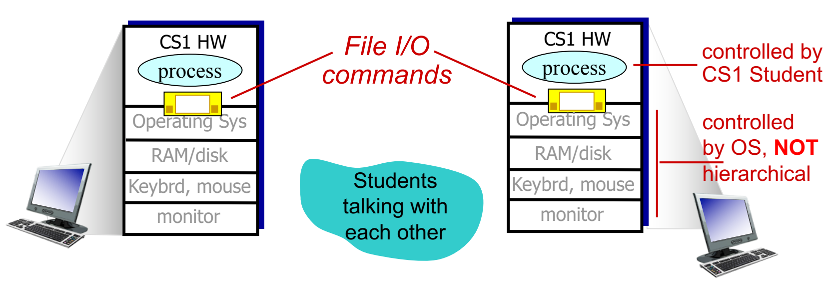
```
# Example of Python file I/O
# read a file back and print all the lines
file = open( "college.txt", "r" )
allLines=file.readlines()  # allLines is a list of strings
file.close()

# your "application" separate from the files
oneString = "" . join( allLines )
Print(repr( oneString )) # repr() makes special chars visible
Print(oneString )        # print it normally
```

33

## File I/O Programming

- Your CS1 program communicates with your computer's operating system to access memory, keyboard input & writing output to the monitor.
- This is an ***approximate*** analogy



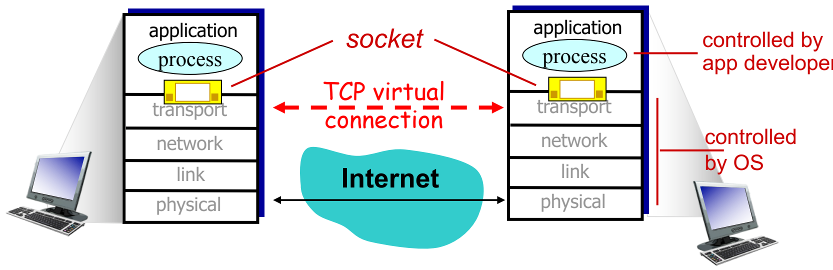

34

# Socket Programming

Application layer communication via the transport layer

*goal:* learn how to build client/server applications that communicate using sockets

*socket:* door between application process and end-end-transport protocol



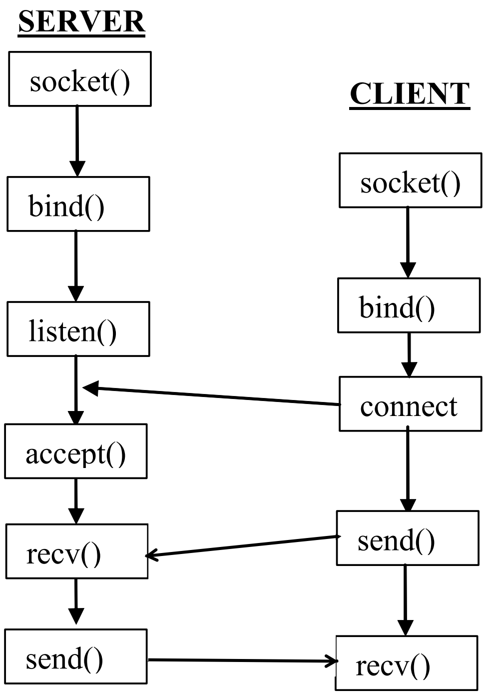

35

# Socket API Overview

- TCP Socket Programming Procedures
  - Socket()
  - Bind()
  - Listen()
  - Accept()
  - Connect()
  - Send and receive procedures
  - Close()
- And for DNS…
  - getHostByName
  - getServByName
  - getProtoByName

36

# TCP Flow Chart




Bind the socket to a port → s.bind((", 80)) specifies that the socket is reachable by any address the machine happens to have

37