Chapter 2
Application Layer

A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application layer: overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application layer: overview

Our goals:

- conceptual *and* implementation aspects of application-layer protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- learn about protocols by examining popular application-layer protocols
  - HTTP
  - SMTP, IMAP
  - DNS
- programming network applications
  - socket API

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Some network apps

- social networking
- Web
- text messaging
- e-mail
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- P2P file sharing
- voice over IP (e.g., Skype)
- real-time video conferencing
- Internet search
- remote login
- …

Q: your favorites?

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Creating a network app

write programs that:

- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

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

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Network Application Architectures

- Client server
- P2P

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Client-server paradigm

server:

- always-on host
- permanent IP address
- often in data centers, for scaling

clients:

- contact, communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do *not* communicate directly with each other
- examples: HTTP, IMAP, FTP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Peer-peer architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- each peer generates workload by requesting files
- each peer also adds service capacity to the system by distributing files to other peers
- peers are intermittently connected and change IP addresses
  - complex management
- example: P2P file sharing
- P2P architectures are also cost effective, since they normally don't require significant server infrastructure and server bandwidth
- P2P applications face challenges of security, performance, and reliability due to their highly decentralized structure.

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Processes communicating

process: program running within a host

- within same host, two processes communicate using inter-process communication (defined by OS)
- processes in different hosts communicate by exchanging messages
  - A sending process creates and sends messages into the network
  - a receiving process receives these messages and
  - possibly responds by sending messages back.

clients, servers

client process: process that initiates communication

server process: process that waits to be contacted

- note: applications with P2P architectures have client processes & server processes

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Sockets

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process pushes message out to door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process
  - two sockets involved: one on each side

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Socket Communication Between Two Processes

Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

- A socket is the interface between the application layer and the transport layer within a host.
- It is also referred to as the Application Programming Interface (API) between the application and the network
- The application developer has control of everything on the application-layer side of the socket but has little control of the transport-layer side of the socket.
  - (1) the choice of transport protocol
  - (2) to fix a transport-layer parameters such as maximum buffer and maximum segment sizes

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
  - *A:* no, *many* processes can be running on same host
- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80
- more shortly…

Copyright © 2022 Pearson India Education Services Pvt. Ltd

An application-layer protocol defines:

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

open protocols

- defined in RFCs, everyone has access to protocol definition
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols

- e.g., Skype

Copyright © 2022 Pearson India Education Services Pvt. Ltd

What transport service does an app need?

data integrity

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Security

To provide confidentiality - Encryption and Decryption

- Data integrity & end-point authentication

throughput

- Because other sessions will be sharing the bandwidth along the network path, and because these other sessions will be coming and going, the available throughput can fluctuate with time.
- Many current multimedia applications are bandwidth sensitive
- other apps ("elastic apps") make use of whatever throughput they get
- bandwidth-sensitive applications have specific throughput requirements
- **elastic applications** can make use of as much, or as little, throughput as happens to be available.

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Transport service requirements: common apps

application	data loss	throughput	time sensitive?
file transfer/download	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	yes, 10's msec
streaming audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	Kbps+	yes, 10's msec
text messaging	no loss	elastic	yes and no

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Network Applications vs Application-layer Protocols

- An application-layer protocol is only one piece of a network application.
- Applications consists of many components including application layer protocol.

Example :

- **WEB**
  - client-server application that allows users to obtain documents from Web servers on demand.
  - consists of many components, including a standard for document formats (that is, HTML), Web browsers, Web servers, and an application-layer protocol.
  - The Web's application-layer protocol, HTTP, defines the format and sequence of messages exchanged between browser and Web server. Thus, HTTP is only one piece of the Web application.
- **Netflix's video service**
  - Components are
    - Servers that store and transmit videos
    - other servers that manage billing and other client functions
    - clients (e.g., the Netflix app on your smartphone, tablet, or computer)
    - an application-level DASH protocol defines the format and sequence of messages exchanged between a Netflix server and client.
  - DASH is only one piece of the Netflix application.

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* regulate sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup.

Q: why bother? *Why* is there a UDP?

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Internet transport protocols services

application	application layer protocol	transport protocol
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
streaming audio/video	HTTP [RFC 7320], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Securing TCP

Vanilla TCP & UDP sockets:

- no encryption
- cleartext passwords sent into socket traverse Internet in cleartext (!)

Transport Layer Security (TLS)

- provides encrypted TCP connections
- data integrity
- end-point authentication

TSL implemented in application layer

- apps use TSL libraries, that use TCP in turn

TLS socket API

- cleartext sent into socket traverse Internet *encrypted*
- see Chapter 8

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application layer: overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Web

- Traditional broadcast - radio and television, which force users to tune in when the content provider makes the content available.
- Web operates *on demand* - Users receive what they want, when they want it.
  - Any individual to make information available over the Web
  - everyone can become a publisher at extremely low cost.
  - Hyperlinks and search engines help us navigate through an ocean of information.

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Web and HTTP

- web page consists of *objects,* each of which can be stored on different Web servers
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of *base HTML-file* which includes *several referenced objects, each* addressable by a *URL,* e.g.,

$$\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\underbrace{\texttt{/someDept/pic.gif}}_{\text{path name}}$$

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP overview

HTTP: hypertext transfer protocol

- HTTP is implemented in two programs: a client program and a server program.
- client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages.
- HTTP defines the structure of these messages and how the client and server exchange the messages
- Web's application layer protocol
- client/server model:
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP overview (continued)

HTTP uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains *no* information about past client requests

aside

protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP connections: two types

Non-persistent HTTP

1. TCP connection opened
2. at most one object sent over TCP connection
3. TCP connection closed

downloading multiple objects required multiple connections

Persistent HTTP

- TCP connection opened to a server
- multiple objects can be sent over *single* TCP connection between client, and that server
- TCP connection closed

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Non-persistent HTTP: example

User enters URL: `www.someSchool.edu/someDepartment/home.index`
(containing text, references to 10 jpeg images)

- 1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80.
- Associated with the TCP connection, there will be a socket at the client and a socket at the server.

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80 "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Non-persistent HTTP: example (cont.)

User enters URL: **`www.someSchool.edu/someDepartment/home.index`**
(containing text, references to 10 jpeg images)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

time

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Non-persistent HTTP: response time

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

HTTP response time (per object):

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- obect/file transmission time

Non-persistent HTTP response time = 2RTT+ file transmission time

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Persistent HTTP (HTTP 1.1)

Non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open multiple parallel TCP connections to fetch referenced objects in parallel

Persistent HTTP (HTTP1.1):

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects (cutting response time in half)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP Request Message

A typical HTTP request message:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close   //after sending the requested object connection
is closed
User-agent: Mozilla/5.0    //browser type
Accept-language: fr      // French version of object is needed
```

- message is written in ordinary ASCII text, so that we can read it
- each line followed by a carriage return and a line feed
- The last line is followed by an additional carriage return and line feed.
- HTTP request message contains :
  - **request line** – first line
    - Fields method field, the URL field, and the HTTP version field
    - The method field including `GET`, `POST`, `HEAD`, `PUT`, and `DELETE`.
    - Mostly, HTTP request messages use the `GET` method.
    - The `GET` method is used when the browser requests an object, with the requested object identified in the URL field.
  - **header lines** - subsequent lines
    - The header line `Host: www.someschool.edu` specifies the host on which the object resides.

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP request message

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)

carriage return character

line-feed character

request line (GET, POST, HEAD commands)

header lines

carriage return, line feed at start of line indicates end of header lines

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP request message: general format

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Other HTTP request messages

POST method:

- web page often includes form input
- user input sent from client to server in entity body of HTTP POST request message
- Bulk messge

GET method (for sending data to server):

- include user data in URL field of HTTP GET request message (following a '?'):

`www.somesite.com/animalsearch?monkeys&banana`

HEAD method:

- requests headers (only) that would be returned *if* specified URL were requested with an HTTP GET method.

PUT method:

- uploads new file (object) to server
- completely replaces file that exists at specified URL with content in entity body of POST HTTP request message

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP response message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html;
   charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP response status codes

- status code appears in 1st line in server-to-client response message.
- some sample codes:

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

- requested object moved, new location specified later in this message (in Location: field in response message)

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet gaia.cs.umass.edu 80
```

- opens TCP connection to port 80 (default HTTP server port) at gaia.cs.umass. edu.
- anything typed in will be sent to port 80 at gaia.cs.umass.edu

2. type in a GET HTTP request:

```
GET /kurose_ross/interactive/index.php HTTP/1.1
Host: gaia.cs.umass.edu
```

- by typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. look at response message sent by HTTP server!

(or use Wireshark to look at captured HTTP request/response)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Maintaining user/server state: cookies

Recall: HTTP GET/response interaction is *stateless*

- no notion of multi-step exchanges of HTTP messages to complete a Web "transaction"
  - no need for client/server to track "state" of multi-step exchange
  - all HTTP requests are independent of each other
  - no need for client/server to "recover" from a partially-completed-but-never-completely-completed transaction

a stateful protocol: client makes two changes to X, or none at all

Q: what happens if network connection or client crashes at *t'* ?

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Maintaining user/server state: cookies

Web sites and client browser use *cookies* to maintain some state between transactions

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, managed by user's browser
4) back-end database at Web site

Example:

- Susan uses browser on laptop, visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID (aka "cookie")
  - entry in backend database for ID
- subsequent HTTP requests from Susan to this site will contain cookie ID value, allowing site to "identify" Susan

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Maintaining user/server state: cookies

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP cookies: comments

What cookies can be used for:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

Challenge: How to keep state:

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: HTTP messages carry state

aside

cookies and privacy:

- cookies permit sites to *learn* a lot about you on their site.
- third party persistent cookies (tracking cookies) allow common identity (cookie value) to be tracked across multiple web sites

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Web caches (proxy servers)

Goal: satisfy client request without involving origin server

- user configures browser to point to a *Web cache*
- browser sends all HTTP requests to cache
  - *if* object in cache: cache returns object to client
  - *else* cache requests object from origin server, caches received object, then returns object to client

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Web caches (proxy servers)

- Web cache acts as both client and server
  - server for original requesting client
  - client to origin server
- typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- reduce response time for client request
  - cache is closer to client
- reduce traffic on an institution's access link
- Internet is dense with caches
  - enables "poor" content providers to more effectively deliver content

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Caching example

Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Average request rate from browsers to origin servers: 15/sec
  - average data rate to browsers: 1.50 Mbps

Performance:

- LAN utilization: .0015
- access link utilization = .97

problem: large delays at high utilization!

- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + minutes + usecs

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Caching example: buy a faster access link

Scenario:

- access link rate: ~~1.54~~ 154 Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Avg request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps

Performance:

- LAN utilization: .0015
- access link utilization = ~~.97~~ .0097
- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + ~~minutes~~ msecs + usecs

Cost: faster access link (expensive!)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Caching example: install a web cache

Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- Web object size: 100K bits
- Avg request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps

Performance:

- LAN utilization: .?
- access link utilization = ?
- average end-end delay = ?

How to compute link utilization, delay?

Cost: web cache (cheap!)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Caching example: install a web cache

Calculating access link utilization, end-end delay with cache:

- suppose cache hit rate is 0.4: 40% requests satisfied at cache, 60% requests satisfied at origin
- access link: 60% of requests use access link
- data rate to browsers over access link
  = 0.6 * 1.50 Mbps = .9 Mbps
- utilization = 0.9/1.54 = .58
- average end-end delay
  = 0.6 * (delay from origin servers)
  + 0.4 * (delay when satisfied at cache)
  = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs

lower average end-end delay than with 154 Mbps link (and cheaper too!)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Conditional GET

Goal: don't send object if cache has up-to-date cached version

- no object transmission delay
- lower link utilization

- *cache:* specify date of cached copy in HTTP request

  If-modified-since: <date>

- *server:* response contains no object if cached copy is up-to-date:

  HTTP/1.0 304 Not Modified

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP/2

Key goal: decreased delay in multi-object HTTP requests

HTTP1.1: introduced multiple, pipelined GETs over single TCP connection

- server responds *in-order* (FCFS: first-come-first-served scheduling) to GET requests
- with FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large object(s)
- loss recovery (retransmitting lost TCP segments) stalls object transmission

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP/2

Key goal: decreased delay in multi-object HTTP requests

HTTP/2: [RFC 7540, 2015] increased flexibility at *server* in sending objects to client:

- methods, status codes, most header fields unchanged from HTTP 1.1
- transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- *push* unrequested objects to client
- divide objects into frames, schedule frames to mitigate HOL blocking

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP/2: mitigating HOL blocking

HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)

objects delivered in order requested: O_2, O_3, O_4 wait behind O_1

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP/2: mitigating HOL blocking

HTTP/2: objects divided into frames, frame transmission interleaved

O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

Copyright © 2022 Pearson India Education Services Pvt. Ltd

HTTP/2 to HTTP/3

Key goal: decreased delay in multi-object HTTP requests

HTTP/2 over single TCP connection means:

- recovery from packet loss still stalls all object transmissions
  - as in HTTP 1.1, browsers have incentive to open multiple parallel TCP connections to reduce stalling, increase overall throughput
- no security over vanilla TCP connection
- HTTP/3: adds security , per object error- and congestion-control (more pipelining) over UDP
  - more on HTTP/3 in transport layer

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application layer: overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

E-mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Outlook, iPhone mail client
- outgoing, incoming messages stored on server

Copyright © 2022 Pearson India Education Services Pvt. Ltd

E-mail: mail servers

mail servers:

- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages
- *SMTP protocol* between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server

Copyright © 2022 Pearson India Education Services Pvt. Ltd

E-mail: the RFC (5321)

- uses TCP to reliably transfer email message from client (mail server initiating connection) to server, port 25
- direct transfer: sending server (acting like client) to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction (like HTTP)
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCI

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Scenario: Alice sends e-mail to Bob

1) Alice uses UA to compose e-mail message "to" bob@someschool.edu

2) Alice's UA sends message to her mail server; message placed in message queue

3) client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Try SMTP interaction for yourself:

telnet <servername> 25

- see 220 reply from server
- enter HELO, MAIL FROM:, RCPT TO:, DATA, QUIT commands

above lets you send email without using e-mail client (reader)

Note: this will only work if <servername> allows telnet connections to port 25 (this is becoming increasingly rare because of security concerns)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

SMTP: closing observations

comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message
- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses CRLF.CRLF to determine end of message

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Mail message format

SMTP: protocol for exchanging e-mail messages, defined in RFC 531 (like HTTP)

RFC 822 defines *syntax* for e-mail message itself (like HTML)

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

  these lines, within the body of the email message area different from SMTP MAIL FROM:, RCPT TO: commands!
- Body: the "message" , ASCII characters only

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Mail access protocols

- SMTP: delivery/storage of e-mail messages to receiver's server
- mail access protocol: retrieval from server
  - IMAP: Internet Mail Access Protocol [RFC 3501]: messages stored on server, IMAP provides retrieval, deletion, folders of stored messages on server
- HTTP: gmail, Hotmail, Yahoo!Mail, etc. provides web-based interface on top of STMP (to send), IMAP (or POP) to retrieve e-mail messages

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application Layer: Overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS: Domain Name System

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., cs.umass.edu - used by humans

Q: how to map between IP address and name, and vice versa ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
  - note: core Internet function, *implemented as application-layer protocol*
  - complexity at network's "edge"

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS: services, structure

DNS services

- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

Q: Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: doesn't scale!

- Comcast DNS servers alone: 600B DNS queries per day

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS: a distributed, hierarchical database

Root

Top Level Domain

Authoritative

Client wants IP address for www.amazon.com; 1st approximation:

- client queries root server to find .com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS: root name servers

- official, contact-of-last-resort by name servers that can not resolve name
- *incredibly important* Internet function
  - Internet couldn't function without it!
  - DNSSEC – provides security (authentication and message integrity)
- ICANN (Internet Corporation for Assigned Names and Numbers) manages root DNS domain

13 logical root name "servers" worldwide each "server" replicated many times (~200 servers in US)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

TLD: authoritative servers

Top-Level Domain (TLD) servers:

- responsible for .com, .org, .net, .edu, .aero, .jobs, .museums, and all top-level country domains, e.g.: .cn, .uk, .fr, .ca, .jp
- Network Solutions: authoritative registry for .com, .net TLD
- Educause: .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

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Local DNS name servers

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called “default 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 may be out of date!)
  - acts as proxy, forwards query into hierarchy

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS name resolution: iterated query

Example: host at engineering.nyu.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"

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Caching, Updating DNS Records

- once (any) 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 servers not often visited
- cached entries may be *out-of-date* (best-effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire!
- update/notify mechanisms proposed IETF standard
  - RFC 2136

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS records

DNS: distributed database storing 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., foo.com)
- `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

- `value` is name of mailserver associated with `name`

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS protocol messages

DNS *query* and *reply* messages, both have same *format:*

message header:

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

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS protocol messages

DNS *query* and *reply* messages, both have same *format:*

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Inserting records into DNS

Example: new startup "Network Utopia"

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts NS, A RRs into .com TLD server:

    `(networkutopia.com, dns1.networkutopia.com, NS)`

    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server locally with IP address `212.212.212.1`
  - type A record for www.networkuptopia.com
  - type MX record for networkutopia.com

Copyright © 2022 Pearson India Education Services Pvt. Ltd

DNS security

DDoS attacks

- bombard root servers with traffic
  - not successful to date
  - traffic filtering
  - local DNS servers cache IPs of TLD servers, allowing root server bypass
- bombard TLD servers
  - potentially more dangerous

Redirect attacks

- man-in-middle
  - intercept DNS queries
- DNS poisoning
  - send bogus relies to DNS server, which caches

Exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification

DNSSEC
[RFC 4033]

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application Layer: Overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Peer-to-peer (P2P) architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, and new service demands
- peers are intermittently connected and change IP addresses
  - complex management
- examples: P2P file sharing (BitTorrent), streaming (KanKan), VoIP (Skype)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

File distribution: client-server vs P2P

Q: how much time to distribute file (size *F*) from one server to *N peers*?

- peer upload/download capacity is limited resource

Copyright © 2022 Pearson India Education Services Pvt. Ltd

File distribution time: client-server

- *server transmission:* must sequentially send (upload) *N* file copies:
  - time to send one copy: F/u_s
  - time to send *N* copies: NF/u_s
- *client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}

time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

increases linearly in N

Copyright © 2022 Pearson India Education Services Pvt. Ltd

File distribution time: P2P

- *server transmission:* must upload at least one copy:
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - min client download time: F/d_{min}
- *clients:* as aggregate must download NF bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$

time to distribute F to N clients using P2P approach

$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in N

… but so does this, as each peer brings service capacity

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$

Copyright © 2022 Pearson India Education Services Pvt. Ltd

P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file

Alice arrives
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

Copyright © 2022 Pearson India Education Services Pvt. Ltd

P2P file distribution: BitTorrent

- peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")

- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn:* peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

Copyright © 2022 Pearson India Education Services Pvt. Ltd

BitTorrent: requesting, sending file chunks

Requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

Sending chunks: tit-for-tat

- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

Copyright © 2022 Pearson India Education Services Pvt. Ltd

BitTorrent: tit-for-tat

(1) Alice “optimistically unchokes” Bob

(2) Alice becomes one of Bob’s top-four providers; Bob reciprocates

(3) Bob becomes one of Alice’s top-four providers

higher upload rate: find better trading partners, get file faster !

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application layer: overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Video Streaming and CDNs: context

- stream video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube, Amazon Prime: 80% of residential ISP traffic (2020)
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution: distributed, application-level infrastructure*

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)

frame *i*

frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Multimedia: video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, 64Kbps – 12 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)

frame *i*

frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Streaming stored video

simple scenario:

Main challenges:

- server-to-client bandwidth will *vary* over time, with changing network congestion levels (in house, in access network, in network core, at video server)
- packet loss and delay due to congestion will delay playout, or result in poor video quality

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Streaming stored video

streaming: at this time, client playing out early part of video, while server still sending later part of video

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Streaming stored video: challenges

- continuous playout constraint: once client playout begins, playback must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match playout requirements
- other challenges:
  - client interactivity: pause, fast-forward, rewind, jump through video
  - video packets may be lost, retransmitted

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Streaming stored video: playout buffering

- *client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks

- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Streaming multimedia: DASH

- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more bandwidth available)
  - *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Streaming video = encoding + DASH + playout buffering

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Content distribution networks (CDNs)

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- option 1: single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Content distribution networks (CDNs)

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
- option 2: store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - Akamai: 240,000 servers deployed in more than 120 countries (2015)
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Content distribution networks (CDNs)

- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy if network path congested

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Content distribution networks (CDNs)

Internet host-host communication as a service

OTT challenges: coping with a congested Internet

- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

Copyright © 2022 Pearson India Education Services Pvt. Ltd

CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B23V

- video stored in CDN at http://KingCDN.com/NetC6y&B23V

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Case study: Netflix

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Application Layer: Overview

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks
- socket programming with UDP and TCP

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Socket programming

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

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

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Socket programming

Two socket types for two transport services:

- *UDP:* unreliable datagram
- *TCP:* reliable, byte stream-oriented

Application Example:

1. client reads a line of characters (data) from its keyboard and sends data to server
2. server receives the data and converts characters to uppercase
3. server sends modified data to client
4. client receives modified data and displays line on its screen

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Socket programming with UDP

UDP: no "connection" between client & server

- no handshaking before sending data
- sender explicitly attaches IP destination address and port # to each packet
- receiver extracts sender IP address and port# from received packet

UDP: transmitted data may be lost or received out-of-order

Application viewpoint:

- UDP provides *unreliable* transfer of groups of bytes ("datagrams") between client and server

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Client/server socket interaction: UDP

server (running on serverIP)

client

create socket, port= x:
serverSocket =
socket(AF_INET,SOCK_DGRAM)

create socket:
clientSocket =
socket(AF_INET,SOCK_DGRAM)

Create datagram with server IP and port=x; send datagram via clientSocket

read datagram from serverSocket

write reply to serverSocket specifying client address, port number

read datagram from clientSocket

close clientSocket

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Example app: UDP client

Python UDPClient

include Python's socket library →

create UDP socket for server →

get user keyboard input →

attach server name, port to message; send into socket →

read reply characters from socket into string →

print out received string and close socket →

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET,
                      SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message.encode(),
                    (serverName, serverPort))
modifiedMessage, serverAddress =
                    clientSocket.recvfrom(2048)
print modifiedMessage.decode()
clientSocket.close()
```

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Example app: UDP server

Python UDPServer

create UDP socket →

bind socket to local port number 12000 →

loop forever →

Read from UDP socket into message, getting client's address (client IP and port) →

send upper case string back to this client →

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print ("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),
                        clientAddress)
```

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Socket programming with TCP

Client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

Client contacts server by:

- Creating TCP socket, specifying IP address, port number of server process
- *when client creates socket:* client TCP establishes connection to server TCP

- when contacted by client, *server TCP creates new socket* for server process to communicate with that particular client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients (more in Chap 3)

Application viewpoint

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Client/server socket interaction: TCP

server (running on hostid)

client

create socket,
port=**x**, for incoming
request:
serverSocket = socket()

wait for incoming
connection request
connectionSocket =
serverSocket.accept()

TCP
connection setup

create socket,
connect to **hostid**, port=**x**
clientSocket = socket()

send request using
clientSocket

read request from
connectionSocket

write reply to
connectionSocket

read reply from
clientSocket

close
connectionSocket

close
clientSocket

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Example app: TCP client

Python TCPClient

create TCP socket for server, remote port 12000

No need to attach server name, port

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence.decode())
clientSocket.close()
```

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Example app: TCP server

Python TCPServer

create TCP welcoming socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new socket created on return

read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while True:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.
                                        encode())
    connectionSocket.close()
```

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Chapter 2: Summary

our study of network application layer is now complete!

- application architectures
  - client-server
  - P2P
- application service requirements:
  - reliability, bandwidth, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP
- specific protocols:
  - HTTP
  - SMTP, IMAP
  - DNS
  - P2P: BitTorrent
- video streaming, CDNs
- socket programming: TCP, UDP sockets

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Chapter 2: Summary

Most importantly: learned about *protocols*!

- typical request/reply message exchange:
  - client requests info or service
  - server responds with data, status code
- message formats:
  - *headers*: fields giving info about data
  - *data:* info(payload) being communicated

important themes:

- centralized vs. decentralized
- stateless vs. stateful
- scalability
- reliable vs. unreliable message transfer
- "complexity at network edge"

Copyright © 2022 Pearson India Education Services Pvt. Ltd

Additional Chapter 2 slides

Copyright © 2022 Pearson India Education Services Pvt. Ltd

1) Suppose within Web browser you click on a link to attain a web page. The IP address for the associated URL is not cached in your local host, so a DNS look up is nessecary to obtain IP address. Suppose that 3 DNS server are visited before your host recieves an IP address from DNS. The First DNS server visited in the local DNS cache with an RTT Delay of $RTT_0 = 1$ millisecond. The Second and Third DNS Server contacted have RTT's of 20 millisecond, 11 millisend. Initially let us suppose that the web page associated with a link contains exactly 1 object consisting of small amount HTML Text. Suppose the RTT between the Localhost and the Webserver containing the object is $RTT_{HTTP} = 65$ millisecond.

a) Assume zero Transmission Time for the HTML object, how much time in millisecond from when the object click on the link until the client recieves the object.

a)

Total time = DNS Query delay + HTTP delay

HTTP delay = 2 * RTT

1 → Connection Setting
Other 1 → getting back response

Total Time = [1 + 20 + 11] + [2 * 65]

= 32 + 130

= 162 ms

b) Now Suppose the HTML Object refrences 8 very small Objects on the Same Server. Neglecting Transmission Time, How much time in millisecond elapses when the Client clicks on the link, until the base object and all 8 additional Objects are recieved from the Webserver and the Client. Assuming non persistent http and no parallel TCP connection.

Total Time = DNS Query + base Object + 8 objects

base Object → 2 * RTT
(HTTP)

8 objects → 8 (2 * RTT)

Here Non-persistent

→ 16 * RTT

if it is persistent only 2 * RTT

= 32 + (2 * 65) + 8 (2 * RTT)

= 32 + 130 + (16 * 65)

= 162 + 1040

= 1202 ms

c) Suppose that HTML object references 8 very small object on the same server, but assume that client is configure to support Max. of 5 parallel TCP connection with non persistent TCP.

Total Time = DNS Query + base object + 8 objects

Here 8 objects, 5 can be parallel hold

First 5 in one parallel connection

remaining 3 in other parallel.

$= 32 + 130 + \underbrace{2 * RTT_{HTTP}}_{\text{first 5}} + \underbrace{2 * RTT_{HTTP}}_{\text{Remaining 3}}$

$= 32 + 130 + 130 + 130$

$= 422$ ms

a) Suppose that HTML object References 8 very small object on the same server, but assume that client is configure to support max. of 5 parallel TCP connection with persistent TCP.

Total time = DNS Query + base object + 8 objects

Here 8 objects in persistent connected, so only one connection to be given. & 5 in parrallel TCP connection.

$1 * RTT_{HTTP} \rightarrow$ First 5

$1 * RTT_{HTTP} \rightarrow$ Remaining 3

Total $\rightarrow 2 * RTT_{HTTP}$ for persistent connection.

$= 32 + 130 + 2 * RTT_{HTTP}$

$= 32 + 130 + 130$

$= 292$ ms

e) Which Method is Faster

persistent with parallel connection is Faster than all other

2) Now Suppose the HTML object References 5 very small objects on the same Server. Neglecting transmission time, how much time (in msec) elapses from when the client clicks on the link until the base objects and all additional 5 objects are recieved from Web server at the client, assuming

i) Non-Persistent HTTP and no parallel TCP Connection.

ii) Non-persistent and parallel connection [5 parallel Tcp Connection is Supported]

iii) Persistent and parallel Connection [5 parallel Tcp Connection is Supported]

APPLICATION LAYER

Text based Applications:

* Text e-mail
* Remote access to computers
* File transfer
* News group and
* Text chat

Killer application of mid 1990s: World wide web, excompanying

* Web surfing
* Search
* e-mail.
* e-commerce.

Audio and video applications:

* Internet Telephony
* Video sharing & streaming
* Internet radio
* IP television

Scanned by TapScanner

APPLICATION LAYER

Text based Applications:

* Text e-mail
* Remote access to computers
* File transfer
* News group and
* Text chat

Killer application of mid 1990s: World wide web, excompanying

* Web surfing
* Search
* e-mail.
* e-commerce.

Audio and video applications:

* Internet Telephony
* Video sharing & streaming
* Internet radio
* IP television

Scanned by TapScanner

Principles of N/w Applications:

① - The core of n/w application development is writing programs that run on different end systems and communicate with each other over the n/w.

- For example in web application, 2 distinct pgms

* Browser pgm running in user's host (desktop, laptop, PDA, cell phone and so on)

* Web server pgm running in the web server host.

② When developing new application, write s/w that will run on multiple end systems.

eg: C, Java

- N/w core devices don't function at the application layer, so, application can't be written for it.

(1) N/w application architecture:

- Application architecture is distinctly different from the n/w architecture.

↓

five layer Internet architecture.

- Application architecture is developed by application

application developer and dictates how the application is structured over the various end systems.

- Two architectural paradigms used in modern n/w applications are

  ★ Client - Server architecture

  ★ Peer - to - Peer architecture

Client Server architecture:

- Server (always - on) host services requests from many other hosts called client.

  - Client host can be either sometimes - on or always - on.

  - eg: web application:

    Web server (Always on) services the request from Browser running on client host.

- When a web server receives a request for an object from a client host, it responds by sending the requested object to the client host.

- In client server architecture, 2 characteristics ① clients do not directly communicate with each other

  For eg: Two browsers don't directly communicate with each other.

  - ② Server has a fixed, well known address called an IP address

Scanned by TapScanner

- Because, the server has fixed, well known address and because the server is always on, a client can always contact the server by sending a packet to the server's address.

eg: Web, FTP, Telnet and e-mail.

- In client-server application, a single server host is incapable of keeping up with all the requests from its clients
- For this reason, a large cluster of hosts - referred to as a data center - is often used to create a powerful virtual server in client-server architecture.
- Application services that are based on client server architecture are often infrastructure intensive

⇓

since they require the service providers to purchase, install and maintain servers.

Scanned by TapScanner

- Additionally, the service providers may pay recurring interconnection and bandwidth costs for sending and receiving data to and from Internet.

eg: Amazon, e-Bay.

P2P architecture:

- There is minimal (or no) reliance on always-on infrastructure servers.
- The application exploits direct communication between pairs of intermittently connected hosts called peers.
- Peers are not owned by the service providers, but are instead desktops and laptops controlled by users.
- Peers communicate without passing through a dedicated server, the architecture is called peer-to-peer.
- More popular and traffic intensive applications are based on P2P architectures.

eg: BitTorrent, eMule, Skype.

Some applications have hybrid architectures, combining both client-server and P2P elements.

Scanned by TapScanner

- For example, Many instant messaging applications, servers are used to track the IP address of users, but user-to-user messages are sent directly between user hosts (without passing through intermediate servers)

- One of the most compelling features of P2P architecture is their self scalability

- P2P architecture are also cost effective, since they don't require significant server infrastructure and bandwidth.

Future P2P applications face three major challenges:

* ISP friendly - Most residential ISPs have been dimensioned for "asymmetrical" bandwidth usage.

* Security - Because of highly distributed and open nature, P2P applications can be a challenge to secure.

* Incentives - P2P applications also depends on convincing users to volunteer B, storage and computation resources to the applications.

Scanned by TapScanner

it works on all platform

Xampp :- Cross platform (X), Apache (A) mysql (M), Php (P) and Perl (P).
→ a local web server.

(2) Processes Communicating:

- A Process is a program that is running within an end system.
- When processes are running on the same end s/m, they communicate with each other, with interprocess Communication, using rules that are governed by the end s/m's operating s/m.
- Processes on two different end systems communicate with each other by exchanging messages across the computer network.
- A sending process creates and sends messages into the n/w; a receiving process receives theses messages and possibly responds by sending messages back.

Scanned by TapScanner

Client and Server Processes:

- A N/w application consists of pair of processes that send msg to each other over a N/w.
  - eg: web application, client browser process exchanges msg with a web server process.
- In some applications, such as P2P file sharing, a process can be both a client and a server.
- The process that initiates the communication session b/w a pair of processes, is labeled as client. The process that waits to be contacted to begin the session is the server.

Interface between the process and the computer N/w:

- Any msg sent from one process to another process must go through the underlying n/w.
- A process sends messages into, and receives messages from, the network through a s/w interface called socket.

eg: home (Process)
door (socket).

Scanned by TapScanner

- Socket is the interface between application & transport layer within a host.
- It is also referred to as Application Programming Inter-face (API) between the application & the n/w, since socket is the programming i/f in which n/w applications are built.
- The application developer can only control on transport layer side
  (i) Choice of transport protocol.
  (ii) Ability to fix a few transport layer parameters such as maximum buffer and maximum segment size.

Transport Services available to applications:

- Many n/w including Internet, provide more than one transport-layer protocol.
- When the application is developed, one of the available transport layer protocols have to be chosen.
- Transport Layer protocol offers the following services such as
  - ★ Reliable data transfer
  - ★ Throughput
  - ★ Timing &
  - ★ Security

Scanned by TapScanner

Reliable data transfer:

- Packets can be lost in computer n/w due to
  - * Buffer overflow in a router
  - * Host or router would have discarded it due to corrupted bits.
- For many applications such as e-mail, file transfer, data loss can have devastating consequences.
- To support these applications, appropriate protocol has been choosen to guarantee that the data sent by one end of the application is delivered correctly and completely to the other end of the application.
- If a protocol provides such a guaranteed data delivery service, it is said to be reliable data transfer.
- Transport layer protocol can potentially provide service to an application is process-to-process $\boxed{\text{reliable}}$ data transfer.

  ↓ Complete confidence that data will arrive w/o errors.
- Loss-tolerant applications (multimedia) applications such as real time audio/video can tolerate some amount of data loss.

throughput:

- Throughput is the rate at which the sending process can deliver bits to the receiving process.

Scanned by TapScanner

- Transport layer can provide guaranteed available thoroughput at specified rate.
- For eg: If application request a guaranteed throughput of 'r' bits/sec then the transport protocol should ensure that available throughput is always atleast 'r' bits/sec
- Applications that have throughput requirements are said to be bandwidth sensitive applications.
- Elastic applications can make use of as much or little throughput as happens to be available.

  eg: e-mail, file transfer.

Timing:

- Transport layer protocol also provides timing guarantees.
- certain applications need tight timing constraints on data delivery in order to be effective.

for eg:
- Long delays in Internet telephony tends to result in unnatural pauses in the conversation.
- For non-real-time applications, lower delay is always preferable to higher delay but no tight constraint is placed on end-to end delays.

Scanned by TapScanner

Security:

- Transport protocol can provide an application with one or more security services such as

* confidentiality
* Data integrity
* end-point authentication.

Scanned by TapScanner

Security:

- Transport protocol can provide an application with one or more security services such as
  * confidentiality
  * Data integrity
  * end-point authentication.

(4) Transport Services Provided by the Internet:

- The Internet makes two transport protocols available to applications
  * UDP and
  * TCP

	Application	Data Loss	Bandwidth	Time-Sensitive
1)	File transfer	No Loss	Elastic	No
2)	E-mail SMTP-TCP	No Loss	Elastic	No
3)	Web documents	No Loss	Elastic (few kbps)	No
4)	Internet telephony/ Video Conferencing	Loss-tolerant	Audio: few kbps-1Mbps video: 10kbps-5Mbps	Yes: 100's of msec
5)	Stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
6)	Interactive games	Loss-tolerant	Few kbps-10kbps	Yes: 100s of msec
7)	Instant messaging	No Loss	Elastic	Yes and no.

TCP Services:

- TCP services include a connection-oriented service and a reliable data transfer service.
- When an application invokes TCP as its transport protocol, the application receives both of these services from TCP.

(i) Connection-oriented service:

- TCP has the client and server exchange transport-layer control information with each other before the application-level messages begin to flow.
- This so-called handshaking procedure alerts the client and server, allowing them to prepare for an onslaught of packets.
- After the handshaking phase, a TCP connection is said to exist between the sockets of two processes.
- The connection is a full-duplex connection in that the two processes can send messages to each other over the connection at the same time.
- When the application finishes sending message, it must tear down the connection.
- The service is reffered to as a "connection-oriented" service rather than a "connection" service because the two processes are connected in a very loose manner.

(ii) Reliable data transfer service:

- The communicating process can rely on TCP to deliver all data sent without error and in proper order.

- When one side of the application passes a stream of bytes into the socket, it can count on TCP to deliver the same stream of bytes to the receiving socket, with no missing or duplicate bytes.

- Real time applications are loss-tolerant and do not need a fully reliable transport services.

⇓

So, developers of real time applications often choose to run their applications over UDP rather than TCP

UDP Services:

- UDP is a light-weight transport protocol, providing minimal services.

- UDP is connectionless (no handshaking)

- UDP provides an unreliable data transfer service (no guarantee that the message will ever reach

the receiving process)

- Messages that do arrive at the receiving process may arrive out of order.
- UDP does not include congestion-control mechanism

Many firewalls are configured to block UDP traffic, designers have increasingly chosen to run multimedia and real-time applications over TCP.

Application	Application Layer Protocol	Underlying Transport Protocol.
E-mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Streaming multimedia	HTTP, RTP	TCP (or) UDP
Internet Telephony	SIP, RTP (or) Proprietary (eg: Skype)	UDP

Addressing Processes

To identify the receiving process, two pieces of information need to be specified

(i) Name or address of the host (IP address)

(ii) An identifier that specifies the receiving process in the destination host. (Port number)

Web server - 80 (port number)

Mail server - 25

(5) Application Layer Protocols:

- An application layer protocol defines how an application's processes, running on different end sys, pass messages to each other.

An application layer protocol defines:

* Types of message exchanges (Eg: Request and Response msg)
* Syntax of various msg types
* Semantics of the fields
  ↓ meaning of the information
* Rules for determining when and how a process sends msg and responds to msgs.

2.2 Web and HTTP

Overview of HTTP:

HTTP - Hyper Text Transfer Protocol.

- HTTP is implemented in two programs:

  * client program
  * server program

  } executing on different end sys, talk to each other by exchanging HTTP messages

- HTTP ⇒ defines the structure of these msg and how the client & server exchange the message.

- Web page (also called document) consists of objects
- objects is simply a file (HTML file, JPEG image, Java applet or video clip) that is addressable by a single URL
- Web pages consists of a base HTML file and several referenced objects.

eg: web page contains HTML text and five JPEG images
totally: webpage has 6 objects

Base HTML file references the other objects in the page with the object's URL.

- Each URL has 2 components: hostname of the server that houses the object and object's path name.

eg:

http://www.someschool.edu/CSE/picture.gif

host name — path name

- Web browsers implement the client side of HTTP
- Web servers implement the server side of HTTP, house web objects, each addressable by a URL.
- HTTP defines how web clients request web pages from web servers and how servers transfer web pages to clients.

- HTTP uses TCP as its underlying protocol (transport layer)
- HTTP client first initiates a TCP connection with the server.
- Once the connection is established, the browser and the server processes access TCP through their socket interface.

- HTTP is said to be a stateless protocol, since HTTP server maintains no information about the client.

Non-Persistent and Persistent Connections:

- Depending on the application and on how the application is being used, the series of requests may be made back-to-back, periodically at regular intervals or intermittently.
- When this client-server interaction is taking place over TCP, the application developer needs to make

an important decision - should

★ each request/response pair be sent over a separate TCP connection (or) (Non-Persistent connection)

★ all of the requests and their corresponding responses be sent over the same TCP connection. (Persistent Connection)

- HTTP can use both non-persistent connections (can be configured) and persistent connections (default)

HTTP with non-persistent connections:

eg: Page consists of base HTML file & 10 JPEG images (all 11 resides on same server)

http://www.someschool.edu/somedept/home.index

- HTTP client process initiates a TCP connection to the server www.someschool.edu on port number 80 (default port number for HTTP). Associated with the TCP connection, there will be a socket at the client and a socket at the server.

- HTTP client sends an HTTP request message to the server via its socket. The request message includes the pathname (/somedept/home.index)

- HTTP Server process receives the request message via its socket, retrieves the object from its (RAM or disk) storage, encapsulates the object in HTTP response

response message and sends the response message to the client via its socket.

- The HTTP server process tells TCP to close the TCP connection
- The HTTP client receives the response message. TCP connection terminates. The msg indicates that the encapsulated file is HTML file. The client extracts the file from the response msg. (examines HTML file and finds reference to 10 JPEG objects)
- First four ~~ets~~ steps are then repeated for each of the referenced JPEG objects.

Steps for non-persistent connection ⇓ each TCP connection transports exactly one request message & one response msg.

- HTTP has nothing to do with how a web page is interpreted by a client.
- Users can configure modern browsers to control the degree of parallelism. =1 ⇒ 10 connections are established serially.
- In their default modes, most browsers open 5 to 10 parallel TCP connections and each of these connections handles one request - response txn.
- Round Trip Time (RTT) is the time it takes for a small packet to travel from client to server and then back to the client.

- RTT includes packet-propagation delay, packet queuing delay in intermediate routers and switches, packet processing delay.

- Browser to initiate TCP connection b/w the browser and the web server

  * This involves 3-way handshake.

Total response time = 2 RTTs + txn time at the server

HTTP with Persistent connections:

* Non-persistent connections have some shortcomings.
* A brand new connection must be established and maintained for each requested object.
* For each of these connections,
    * TCP buffers must be allocated
    * TCP variables must be kept in both client & server.

disadvantage: A significant burden on web server.
Each object suffers a delivery delay of Two RTTS (1 RTT to establish TCP Connection & 1 RTT to request & receive an object)

* With persistent connections, the server leaves the TCP connection open after sending a response
* Subsequent requests & responses between the same client & server can be sent over the same connection.
* In particular, an entire web page (base HTML file & 10 images) can be sent over a single persistent connection.
* Multiple web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection.

- The default mode of HTTP uses persistent connection with pipelining.

HTTP Message Format:

Mail Message Formats:

- When an email message is sent from one peer to another, a header containing peripheral information is contained in a series of header lines.
- The header lines and the body are separated by blank line.
- Each header line contains readable text, consisting of a keyword followed by a colon followed by value.
- Some of the keywords are required and others are optional.

From:
To:
Subject:

- The SMTP commands are different from these header lines.
- SMTP commands are part of SMTP handshaking protocol and the headerlines are part of mail

Scanned by TapScanner

- A typical message header look like this:

From: a@crepes.fr

To: b@bamburger.edu

Subject: Searching for the meaning of life.

- After the message header, a blank line follows then the message body follows.
- To send a msg to mail server it contains some headerlines including the Subject: header line.
- To do this, issue telnet servername 25

Mail Access Protocols:

- Today, mail access uses a client-server architecture - the user reads e-mail with a client that executes on the user's end sys.

Eg: on an office PC, laptop or smartphone.

- The ~~user agent~~ recipient executes his user agent on his local PC, it is natural to consider placing a mail server on his local PC as well.

⇓

Sender's mail server would dialogue directly with Recipient's PC.

Recall: Mail server manages mailboxes and runs the client and server sides of SMTP.

Scanned by TapScanner

- The recipient's PC would have to remain always on and connected to the Internet, in order to receive new mail, which can arrive at any time.
- This is impractical for many Internet users.

Obtaining mail is a pull operation but SMTP is a push protocol

So, (POP3), IMAP and HTTP
Popular mail access protocol.

POP3:

- POP3 is an extremely simple mail access protocol.
- POP3 begins when the user agent opens a TCP connection to the mail server on port 110
- With the TCP connection established, POP3 progress through three phases:

  * Authorisation - User sends a username & pwd to authenticate the user
  * Transaction & User agent retrieves message and
  * Update the user agent can mark messages for deletion, remove deletion marks and obtain mail statistics.

  → This phase occurs at the client has issued quit command ending the POP3 session. The mail server deletes the message

that were marked for deletion.

- In a POP3 transaction, the user agent issues commands & the server responds to each with a reply. There are two possible responses:

  OK - Server indicates that the previous command was fine.

  ERR - Server to indicate that something was wrong with previous command.

- The authorisation phase has 2 principal commands:

  user<username>

  pass<password>

- The transaction phase: A user agent using POP3 can be configured to

  * download and delete - list, retr and delete commands.
  * download and keep.

eg: <u>download and delete</u>:

C: list

S: 1 498

S: 2 912

S: .

C: retr 1

S: (.....

S: .....

S: .....)

Scanned by TapScanner

S: .

C: dele 1

C: retr 2

S: (blah blah ....

S: .....

S: .....)

S: .

C: dele 2

C: quit

S: +OK POP3 server signing off

After processing the quit command, POP3 enters the update phase and removes messages 1 and 2 from the mail box

Problem in download & delete mode

* If recipient first reads a message on his office PC, he will not be able to reread the message from his portable at home later in the evening.
* In download and keep mode, the user agent leaves the messages on the mail server after downloading them.
* In this case, Bob can reread messages from different machines
* During a POP3 session between a user agent &

Scanned by TapScanner

the mail server, the POP3 server maintains some state information, in particular, it keeps track of which user messages have been marked deleted.

- POP3 server does not carry state information across POP3 session. (stateless protocol)

IMAP:

- With POP3 access, once the message has been downloaded to the local machine, mail folders has been created and the downloaded messages can be moved to that folder.
- Folders and messages works fine in the local machine but in remote server it causes the problem.
- This is not possible with POP3 - the POP3 protocol does not provide any means for a user to create remote folders and assign messages to folders.
- To solve this issue, IMAP protocol can be used.
- IMAP is a mail access protocol.
- It has many more features than POP3, but it also significantly more complex.
- An IMAP server will associate each message with a folder, when a msg first arrives at the server, it is associated with recipient's INBOX folder.

Scanned by TapScanner

- The recipient can then move the message into a new, user-created folder, read the message, delete the message and so on.

- IMAP protocol provides command to allow users to create folders and move messages from one folder to another.

- IMAP also provides commands that allow users to search remote folders for messages matching specific criteria.

- IMAP server maintains $\boxed{\text{user state information}}$ across IMAP sessions - eg: Names of the folders and which messages are associated with which folders.

- Another feature of IMAP is that it has commands that permit a user agent to $\boxed{\text{obtain components of msg}}$

eg: A useragent can obtain just the message header of a msg or just one part of a multipart MIME message

⇓

useful for low-bandwidth connection

Web-based E-mail:

- Now, web based e-mail is provided by Google, Yahoo as well as just about every major university & corporation.

Scanned by TapScanner

- With this service, the user agent is an ordinary web browser and the user communicates with its remote mailbox via HTTP
- when a recipient wants to access a message in his mailbox, email msg is sent from mail server to the browser using HTTP protocol
- when a sender wants to send an e-mail message, the email message is sent from her browser to her mail server over HTTP rather than SMTP.
- Sender's mail server still sends msg to and receive msg from, other mail servers using SMTP.

Scanned by TapScanner

2.2 Web and HTTP

Overview of HTTP:

HTTP - Hyper Text Transfer Protocol.

- HTTP is implemented in two programs:

  ★ client program
  ★ server program

  } executing on different end sys, talk to each other by exchanging HTTP messages

- HTTP ⇒ defines the structure of these msg and how the client & server exchange the message.

Scanned by TapScanner

- Web page (also called document) consists of objects
- objects is simply a file (HTML file, JPEG image, Java applet or video clip) that is addressable by a single URL
- Web pages consists of a base HTML file and several referenced objects.

eg: web page contains HTML text and five JPEG images
Totally: webpage has 6 objects

Base HTML file references the other objects in the page with the object's URL.

- Each URL has 2 components: hostname of the server that houses the object and object's path name.

eg:
http://www.someschool.edu/CSE/picture.gif
host name path name

- Web browsers implement the client side of HTTP
- Web servers implement the server side of HTTP, house web objects, each addressable by a URL.
- HTTP defines how web clients request web pages from web servers and how servers transfer web pages to clients.

Scanned by TapScanner

- HTTP uses TCP as its underlying protocol (transport layer)
- HTTP client first initiates a TCP connection with the server.
- Once the connection is established, the browser and the server processes access TCP through their socket interface.

- HTTP is said to be a stateless protocol, since HTTP server maintains no information about the client.

Non-Persistent and Persistent Connections:

- Depending on the application and on how the application is being used, the series of requests may be made back-to-back, periodically at regular intervals or intermittently.
- When this client-server interaction is taking place over TCP, the application developer needs to make

an important decision - should

★ each request/response pair be sent over a separate TCP connection (or) (Non - Persistent connection)

★ all of the requests and their corresponding responses be sent over the same TCP connection. (Persistent Connection)

- HTTP can use both non-persistent connections (can be configured) and persistent connections (default)

HTTP with non-persistent Connections:

eg: Page consists of base HTML file & 10 JPEG images (all 11 resides on same server)

http://www.someschool.edu/somedept/home.index

- HTTP client process initiates a TCP connection to the server www.someschool.edu on port number 80 (default port number for HTTP). Associated with the TCP connection, there will be a socket at the client and a socket at the server.

- HTTP client sends an HTTP request message to the server via its socket. The request message includes the pathname (/somedept/home.index)

- HTTP Server process receives the request message via its socket, retrieves the object from its (RAM or disk) storage, encapsulates the object in HTTP response

response message and sends the response message to the client via its socket.

- The HTTP server process tells TCP to close the TCP connection

- The HTTP client receives the response message. TCP connection terminates. The msg indicates that the encapsulated file is HTML file. The client extracts the file from the response msg. (examines HTML file and finds reference to 10 JPEG objects)

steps for non-persistent connection ⇓ each TCP connection transports exactly one request message & one response msg.

- First four ~~etc~~ steps are then repeated for each of the referenced JPEG objects.

- HTTP has nothing to do with how a web page is interpreted by a client.

- Users can configure modern browsers to control the degree of parallelism. =1 ⇒ 10 connections are established serially.

- In their default modes, most browsers open 5 to 10 parallel TCP connections and each of these connections handles one request-response txn.

- Round Trip Time (RTT) is the time it takes for a small packet to travel from client to server and then back to the client.

Scanned by TapScanner

- RTT includes packet - propagation delay, packet queuing delay in intermediate routers and switches, packet processing delay.

- Browser to initiate TCP connection b/w the browser and the web server

  * This involves 3-way handshake.

$$\boxed{\text{Total response time} = 2\ \text{RTTs} + \text{txn time at the server}}$$

Scanned by TapScanner

HTTP with Persistent connections:

* Non-persistent connections have some shortcomings.
* A brand new connection must be established and maintained for each requested object.
* For each of these connections,
    * TCP buffers must be allocated
    * TCP variables must be kept in both client & server.

disadvantage: A significant burden on web server.
Each object suffers a delivery delay of Two RTTS (1 RTT to establish TCP connection & 1 RTT to request & receive an object)

* With persistent connections, the server leaves the TCP connection open after sending a response
* Subsequent requests & responses between the same client & server can be sent over the same connection.
* In particular, an entire web page (base HTML file & 10 images) can be sent over a single persistent connection.
* Multiple web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection.

Scanned by TapScanner

- The default mode of HTTP uses persistent connection with pipelining.

HTTP Message Format.

There are two types of HTTP messages

★ Request message

★ Response message.

- The first line of an HTTP request message is called the request line and the subsequent lines are called the header lines.

- The request line has three fields: the method, the URL and HTTP version field.

- The method can take on different values including GET, POST, HEAD, PUT and DELETE.

- The great majority of HTTP request message use GET method.

- The GET method is used when the browser requests an object, with the requested object identified in the URI field.

eg: If browser is requesting for an object

/somedir/page.html

Browser implements the version HTTP/1.1

Scanned by TapScanner

specifies the host on which object resides

Host: www.someschool.edu

The browser is telling the server that it doesn't want to bother with persistent connections

Connection: close

browser type that is making the request to the server.

User-agent: Mozilla/5.0

User prefers to receive a French version of the object

Accept-Language: fr

otherwise default version is sent

This headerline is useful becoz the server can actually send different versions of the same object to different types of user agents.

General format of HTTP request message

- After the header line, there is an "entire body".
- HTML form often use the GET method and include the inputted data in the requested URL.

eg: www.somesite.com/animalsearch?monkeys+bananas

inputted data.

Scanned by TapScanner

- HEAD method is similar to GET method.
- Application developers often use the HEAD method for debugging.
- The PUT method is often used in conjunction with web publishing tools.
- The PUT method is also used by applications that need to upload objects to web servers.
- The DELETE method allows a user or an application, to delete an object on a web server.

HTTP Response Message:

- The response message has three sections:
  - ★ Status line
  - ★ Six header line and
  - ★ entire body.
- The entire body contains the requested object itself.
- The status line has three fields:
  - ★ The protocol version
  - ★ Status code
  - ★ Corresponding status msg.

eg: HTTP/1.1 200 OK

Server is using HTTP/1.1

Scanned by TapScanner

Header lines:

eg:

Connection: close → to tell the client that is going to close TCP connection after sending msg

Not the time when the object was created or last modified

Date: Tue, 09 Aug 11 15:44:04 GMT → time & date when HTTP response was created & sent by the server

Server: Apache/2.2.3 (CentOS)

Last Modified: Tue, 09 Aug 2011 15:11:03 GMT

indicate time & date when the objected was created or last modified

Content-length: 6821 // Number of bytes in object being sent

Content-Type: text/html // Object is the entire body is HTML text.

(data data data data...)

Some status codes are:

200 - OK - Request succeeded and the information is returned in the response.

400 - Bad Request - The request could not be understood by the server

404 - Not found - The requested document doesn't exist on this server.

505 - HTTP version not supported - The requested HTTP protocol is not supported by server

Scanned by TapScanner

How does a browser decide which header lines to include in a request message? How does a web server decide which header lines to include in a response message?

A browser will generate header lines as a function of the browser type and version, the user configuration of the browser and whether the browser has a cached but possibly out-of-date, version of the object.

Web server behave III^ably: There are different products, versions and configurations, all which influence the header lines to be included in the response message.

General format of an HTTP response message

Scanned by TapScanner

User-Server Interaction: Cookies

- Cookies allow sites to keep track of users.
- Cookie technology has four components:
  (1) A cookie header line in HTTP response message
  (2) A cookie header line in HTTP request message
  (3) A cookie file kept on the user's end sys and managed by user's browser.
  (4) A back-end database at the website.

eg:

Scanned by TapScanner

- Ranjith contacts amazon.com for the first time.
- In the past, he has already visited ebay site.
- When the request comes into the Amazon web server, the server creates a unique identification number and creates an entry in its back-end database that is indexed by the identification number.
- The Amazon web server then responds to Ranjith's browser, including in the HTTP response a Set-cookie: header, which contains the identification number.

Set-cookie: 1678

- The browser then appends a line to the special cookie file that it manages.

↓

has already an entry for ebay.

- As Ranjith continues to browse the Amazon site, each time he requests a web page, his browser consults his cookie file, extracts his identification number for this site and puts a [Cookie header line] that includes the identification number in the HTTP request.
- Amazon uses cookies to provide its shopping cart service

Scanned by TapScanner

- Cookies can be used to identify a user.
- The first time a user visits a site, the user can provide a user identification.
- During the subsequent sessions, the browser passes a cookie header to the server, thereby identifying the user to the server.
- Although cookies often simplify the Internet shopping experience for the user, they are controversial because they can also be considered as an invasion of privacy

Web Caching:

- A web cache (also called proxy server) is a network entity that satisfies HTTP request on behalf of an origin web server.
- Web cache has its own disk storage and keeps copies of recently requested objects in this storage.

Scanned by TapScanner

- The User's browser can be configured so that all the user's HTTP requests are first directed to the web cache.
- The steps are

* The browser establishes a TCP connection to the web cache & sends an HTTP request for the object to the web cache.
* The web cache checks to see if it has a copy of the object stored locally. If it does, the web cache returns an object within the HTTP response message to the client browser.
* If the web cache does not have the object, the web cache opens a TCP connection to the origin server. The web cache then sends an HTTP request for the object into the cache-to-server TCP connection. After receiving this request, the origin server sends an object within an HTTP response to the web cache.
* When the web cache receives the object, it stores a copy in its local storage & sends a copy within an HTTP response message to the client browser.

Cache is both a server and a client at the same time

when it receives requests from & responses to a browser

When it sends requests to & receives responses from the Origin server.

Scanned by TapScanner

- Web cache is purchased and installed by an ISP.
- Web caching has seen deployment in the Internet for two reasons:
  - ☆ Web cache can substantially reduce the response time for a client request.
  - ☆ Web cache can substantially reduce traffic on an institution's access link to the Internet.

⇓

No need to upgrade B

⇓

Reduces costs.

- The above figure has 2 n/ws: Institual n/w and the rest of public Internet.

  Institual n/w → is a high speed LAN

- Router in the institutional N/w & a router in the Internet are connected by 15Mbps

- If the avg object size is 1Mbits and that the avg request from Institution browser to the Origin server is 15 requests per second. ⇒ If HTTP request msgs are negligibly small, it creates no traffic in the N/ws or in the access link.

- The amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response (Internet delay)

Total response time = Time from browser's request of an object until its receipt of the object.

= LAN delay + Access delay + Internet delay

Traffic Intensity on LAN = $\frac{(15 \text{ req/sec}) \times (1 \text{ Mbits/req})}{100 \text{ Mbps}} = 0.15$

↓ very small (negligible)

Traffic Intensity on access link = $\frac{(15 \text{ req/sec}) \times (1 \text{ Mbits/req})}{15 \text{ Mbps}}$

$= 1$

As the traffic intensity approaches 1, the delay on a link becomes very large and grows without bound.

Scanned by TapScanner

- Hit rate - The fraction of requests that are satisfied by a cache.

* Ranges from 0.2 to 0.7 in practice.

For eg:

Instead of upgrading the accessing link, installing web cache in the institutional N/w is an alternative solution.

Hit rate - the fraction of requests that are satisfied by a cache

* It ranges from 0.2 to 0.7

* If cache provides a hit rate of 0.4 for this institution. (40% of requests are satisfied almost immediately within 10 milliseconds by the cache)

* 60% of requests are satisfied by the Origin Server.

The Conditional GET

- The copy of the object residing in the cache may be stale.
- HTTP has a mechanism that allows a cache to verify that its objects are upto date.

Scanned by TapScanner

- This mechanism is called the conditional GET.
- An HTTP request message is a so-called conditional GET msg if (1) the request message uses the GET method and (2) the request message includes If-Modified-Since: header line.

Operation of conditional GET:

- On behalf of a requesting browser, a proxy cache sends a request message to a web server
  - GET /fruit/kiwi.gif HTTP/1.1
    Host: www.exotiquecuisine.com
- The web server sends a response msg with the requested object to the cache.
  HTTP/1.1 200 OK
  Date: Sat, 7 Jul 2007 15:39:29
  Server: Apache/1.3.0 (Unix)
  Last-Modified: Wed, 4 Jul 2007 09:23:24
  (data data data data data ...)
- The cache finds the object to the requesting browser & also caches the object locally.
- After a week, another browser requests the same object via the cache & the object is still in the cache.

Scanned by TapScanner

- Since this object may have been modified at the web server in the past week, the cache performs an up-to-date check by issuing a conditional GET.

GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 4 July 2007 09:23:24

- The conditional GET is telling the server to send the object only if the object has been modified since the specified date.
- Webserver sends a response message but doesn't include ~~HTTP~~ the requested object in the response msg.

status code.

Saves β
↑es user-preceived response time.

HTTP/1.1 304 Not modified
Date: Sat, 14 July 2007 15:39:29
Server: Apache/1.3.0 (Unix)

(empty entire body).

Scanned by TapScanner

Electronic Mail in the Internet:

Internet Mail s/m has 3 major components:

- ★ User agent
- ★ Mail servers and
- ★ Simple Mail Transfer protocol.

Scanned by TapScanner

~~Simple Mo~~

Alice

↓

User agents allows user to read, reply to, fwd, save & compose msg

Eg: Microsoft Outlook
of user agent Apple Mail

↓

User agent sends the msg

Bob

↓ wants to access msg in his mailbox

Authenticates

User agent retrives the message

Bob's Mail box

Mail Server

(Mail placed in mail server's outgoing message queue)

Sender (Alice)

* Couldn't deliver mail to Bob's Server, Alice server holds the message in a msg queue & attempts to transfer the msg later.

* Each recipient has a mailbox located in one of the mail servers.

* Reattempts are often done every 30 minutes or so; if there is no success after several days, the server removes the message and notifies the sender with an e-mail msg.

Scanned by TapScanner

- SMTP is the principal application layer protocol for Internet electronic mail.
- It uses the reliable data transfer service of TCP to transfer mail from sender's mail server to the recepient's mail server.
- SMTP has two sides:

  A client side: which executes on sender's mail server

  Server side: Executes on recipient's mail server.
- Both client and server sides of SMTP run on every mail server.
- When a mail server sends mail to other mail servers, it acts as an SMTP client.
- When a mail server receives mail from other mail servers, it acts as an SMTP server.

SMTP:

- SMTP is much older than HTTP

Scanned by TapScanner

- It is important to observe that SMTP does not normally use intermediate mail servers for sending mail.

eg. Alice's server (Hong Kong) TCP is a direct connection Bob's server (St. Louis)

(If bob's server is down, msg remains in Alice's server & waits for a new attempt.

Msg does not get placed in intermediate mail server

① Client SMTP has TCP establish a connection to port 25 at the server SMTP

② If the server is down, the client tries again later.

③ Once the connection is established, the server and the client perform some application-layer handshaking.

④ During this SMTP handshaking phase, the SMTP client indicates the e-mail address of the sender & e-mail address of the recipient.

⑤ Once the SMTP client and server have introduced themselves to each other, the client sends the message.

⑥ SMTP can count on reliable data transfer service of TCP to get the message to the server without errors.

⑦ The client then repeats this process over the same TCP connection if it has other messages to send to the server; otherwise it instructs TCP to close the connection.

Scanned by TapScanner

hostname

Five cmds:
HELO
MAIL FROM
RCPT TO
DATA
QUIT

S: 220 hamburger.edu

C: HELO crepes.fr hostname of client

S: 250 Hello crepes.fr, pleased to meet you

C: MAIL FROM: <alice@crepes.fr>

S: 250 alice@crepes.fr... Sender ok

C: RCPT TO: <bob@hamburger.edu>

S: 250 bob@hamburger.edu .... Recipient ok

C: DATA

S: 354 Enter mail, end with "." on a line by itself

C: Do you like ketchup?
C: How about pickles? } Persistent connection

C: . → end of the msg to the server

S: 250 Message accepted for delivery

C: QUIT

S: 221 hamburger.edu closing connection.

* Server issues replies to each command with each reply having a reply code and some (optional) English-language explanation.

Telnet can be used to carry out a direct dialogue with an SMTP server.

telnet servername 25
↳ can be a local server name

Scanned by TapScanner

Comparison with HTTP

* HTTP transfer files (also called objects) from a web server to a web client.
* SMTP transfers files from one mail server to another mail server.
* When transferring files, both persistent HTTP and SMTP use persistent connections.

Differences:

- HTTP is mainly a pull protocol - Users use HTTP to pull the information from the server and TCP connection is initiated by the machine that wants to receive the file.
- SMTP is a push protocol - The sending mail server pushes the file to the receiving mail server and TCP connection is initiated by the machine that wants to receive the file.
- SMTP requires each message, including the body of each message to be in 7-bit ASCII format.
- If the msg contains characters that are not 7-bit ASCII (French character with accents) or contains binary data (image) then the message has to be encoded

Scanned by TapScanner

into 7-bit ASCII.

- HTTP does not impose this restriction.
- HTTP encapsulates each object in its own HTTP response message.
- Internet mail places all the msg's object into one message.

Scanned by TapScanner

Mail Message Formats:

- When an email message is sent from one peer to another, a header containing peripheral information is contained in a series of header lines.
- The header lines and the body are separated by blank line.
- Each header line contains readable text, consisting of a keyword followed by a colon followed by value.
- Some of the keywords are required and others are optional.

From:
To:
Subject:

- The SMTP commands are different from these header lines.
- SMTP commands are part of SMTP handshaking protocol and the headerlines are part of mail

Scanned by TapScanner

- A typical message header look like this:

From: a@crepes.fr

To: b@bamburger.edu

Subject: Searching for the meaning of life.

- After the message header, a blank line follows then the message body follows.
- To send a msg to mail server it contains some headerlines including the Subject: header line.
- To do this, issue telnet servername 25

Mail Access Protocols:

- Today, mail access uses a client-server architecture - the user reads e-mail with a client that executes on the user's end sys.

Eg: on an office PC, laptop or smartphone.

- The ~~user agent~~ recipient executes his user agent on his local PC, it is natural to consider placing a mail server on his local PC as well.

⇓

Sender's mail server would dialogue directly with Recipient's PC.

Recall: Mail server manages mailboxes and runs the client and server sides of SMTP.

Scanned by TapScanner

- The recipient's PC would have to remain always on and connected to the Internet, in order to receive new mail, which can arrive at any time.
- This is impractical for many Internet users.

Obtaining mail is a pull operation but SMTP is a push protocol

So, (POP3), IMAP and HTTP
Popular mail access protocol.

POP3:

- POP3 is an extremely simple mail access protocol.
- POP3 begins when the user agent opens a TCP connection to the mail server on port 110
- With the TCP connection established, POP3 progress through three phases:

  * Authorisation - User sends a username & pwd to authenticate the user
  * Transaction & User agent retrieves message and
  * Update the user agent can mark messages for deletion, remove deletion marks and obtain mail statistics.

  → This phase occurs at the client has issued quit command ending the POP3 session. The mail server deletes the message

that were marked for deletion.

- In a POP3 transaction, the user agent issues commands & the server responds to each with a reply. There are two possible responses:

  OK - Server indicates that the previous command was fine.

  ERR - Server to indicate that something was wrong with previous command.

- The authorisation phase has 2 principal commands:

  user<username>

  pass<password>

- The transaction phase: A user agent using POP3 can be configured to

  * download and delete - list, retr and delete commands.
  * download and keep.

eg: <u>download and delete</u>:

C: list

S: 1 498

S: 2 912

S: .

C: retr 1

S: (.....

S: .....

S: .....)

Scanned by TapScanner

S: .

C: dele 1

C: retr 2

S: (blah blah ....

S: .....

S: .....)

S: .

C: dele 2

C: quit

S: +OK POP3 server signing off

After processing the quit command, POP3 enters the update phase and removes messages 1 and 2 from the mail box

Problem in download & delete mode

* If recipient first reads a message on his office PC, he will not be able to reread the message from his portable at home later in the evening.
* In download and keep mode, the user agent leaves the messages on the mail server after downloading them.
* In this case, Bob can reread messages from different machines
* During a POP3 session between a user agent &

Scanned by TapScanner

the mail server, the POP3 server maintains some state information, in particular, it keeps track of which user messages have been marked deleted.

- POP3 server does not carry state information across POP3 session. (stateless protocol)

IMAP:

- With POP3 access, once the message has been downloaded to the local machine, mail folders has been created and the downloaded messages can be moved to that folder.
- Folders and messages works fine in the local machine but in remote server it causes the problem.
- This is not possible with POP3 - the POP3 protocol does not provide any means for a user to create remote folders and assign messages to folders.
- To solve this issue, IMAP protocol can be used.
- IMAP is a mail access protocol.
- It has many more features than POP3, but it also significantly more complex.
- An IMAP server will associate each message with a folder, when a msg first arrives at the server, it is associated with recipient's INBOX folder.

Scanned by TapScanner

- The recipient can then move the message into a new, user-created folder, read the message, delete the message and so on.

- IMAP protocol provides command to allow users to create folders and move messages from one folder to another.

- IMAP also provides commands that allow users to search remote folders for messages matching specific criteria.

- IMAP server maintains $\boxed{\text{user state information}}$ across IMAP sessions - eg: Names of the folders and which messages are associated with which folders.

- Another feature of IMAP is that it has commands that permit a user agent to $\boxed{\text{obtain components of msg}}$

eg: A useragent can obtain just the message header of a msg or just one part of a multipart MIME message

⇓

useful for low-bandwidth connection

Web-based E-mail:

- Now, web based e-mail is provided by Google, Yahoo as well as just about every major university & corporation.

Scanned by TapScanner

- With this service, the user agent is an ordinary web browser and the user communicates with its remote mailbox via HTTP
- when a recipient wants to access a message in his mailbox, email msg is sent from mail server to the browser using HTTP protocol
- when a sender wants to send an e-mail message, the email message is sent from her browser to her mail server over HTTP rather than SMTP.
- Sender's mail server still sends msg to and receive msg from, other mail servers using SMTP.

Scanned by TapScanner

DNS – The Internet's Directory Service

- One identifier for a host is its hostnames.
- Host names - such as www.google.com, kongu.edu are mnemonics

- Hostnames provide little, if any, information bout the location within the Internet of the host

  eg: kongu.ac.in
  ↓
  India

- Hostname can consist of variable length alphanumeric characters, they would be difficult to process by routers

Scanned by TapScanner

- For these reasons, hosts are identified by IP address.

Services Provided by DNS:

- A host can be identified by
  - ☆ host name
  - ☆ IP address
- Domain name system (DNS) is (i) distributed database implemented in a hierarchy of DNS servers & (ii) Application layer protocol that allows hosts to query the distributed database.
- DNS servers are often UNIX machines running the Berkeley Internet Name Domain (BIND) software.
- DNS protocol runs over UDP and uses port 53.
- DNS is commonly employed by other application layer protocols - including HTTP, SMTP and FTP - to translate user-supplied hostnames to IP addresses.

www.someschool.edu

* The user machine runs the client side of the DNS application
* The browser extracts the hostname, www.someschool.edu from URL and passes the hostname to the client

Scanned by TapScanner

side of the DNS application.

* DNS client sends a query containing the hostname to a DNS server.
* DNS client eventually receives a reply, which includes the IP address of the hostname.
* Once the browser receives the IP address from DNS, it can initiate a TCP connection to HTTP server process located at port 80 at that IP address.

- DNS adds an additional delay to the Internet applications that uses it.
- The desired IP address is often cached in a "nearby" DNS server, which helps to reduce DNS n/w traffic as well as average DNS delay.

DNS provide a few other important services in addition to translate hostname to IP addresses:

Host aliasing:

* A host with a complicated host name can have one or more alias names.

For eg: a hostname such as relay1.west-coast.enterprise.com have two aliases such as enterprise.com

canonical name

and www.enterprise.com

- The Alias hostnames, when present, are typically more mnemonic than canonical hostnames.
- DNS can be invoked by an application to obtain the canonical hostname for a supplied alias hostname as well as the IP address of the host.

Mail Server aliasing:

- It is highly desirable that e-mail addresses be mnemonic.

Bob - bob@hotmail.com ⇒ canonical host name might be relay1.west-coast.hotmail.com

(bob@hotmail.com: simple)

- DNS can be invoked by a mail application to obtain the canonical hostname for a supplied alias host name as well as IP address of the host.

Load distribution:

- DNS is also used to perform load distribution among replicated servers, such as replicated Web servers.
- Busy sites such as cnn.com are replicated over multiple servers, with each server running on a different end sys and each having a different IP address.

Scanned by TapScanner

- For replicated web servers, a set of IP address is thus associated with one canonical hostname.
- DNS dB contains the set of IP addresses
- When a client makes a DNS query for a name mapped to a set of addresses, the server responds with the entire set of IP addresses, but rotates the ordering of the addresses within each reply.
- Because a client typically sends its HTTP request message to the IP address that is listed first in the set, DNS rotation distributes the traffic among the replicated servers.

Working of DNS:

- Some application running in a user's host needs to translate a hostname to an IP address.
- The application will invoke the client side of DNS, specifies the hostname that needs to be addressed
- On UNIX based m/c, gethostbyname() is the function call that an application calls in order to perform translation.
- DNS in the user's host then takes over, sending a

Scanned by TapScanner

query message into the N/w.

- All DNS query and reply messages are sent within IP ~~address~~ UDP datagrams to port 53.
- After a delay, ranging from milliseconds to seconds, DNS in the user's host receives a DNS reply message that provides the desired mapping.
- Simple design for DNS: 1 DNS server that contains all the mapping. (centralized design)
- Client simply direct all queries to the single DNS server and the DNS server responds directly to all queries to all the ~~single DNS~~ ~~server~~ clients.
- The problem with a centralized design include:

* A single point of failure: If the DNS server crashes, so does the entire Internet!
* Traffic volume: A single DNS server would have to handle all DNS queries
* Distant centralized database: A single DNS server can't be "close to" all the querying clients. If single DNS server is in NewYork then all the queries from Australia must travel to the other side of the globe. This can lead to the significant delays
* Maintenance: The single DNS server would have to keep records

Scanned by TapScanner

for all Internet host. Not only would this centralized db be huge, but it would have to be updated frequently to account for every new host.

- A centralized dB in a single DNS server simply doesn't scale.
- DNS is distributed by design.

Scanned by TapScanner