Computer Networks
Lecture 5: Application Layer

Dr. Hossam Mahmoud Moftah

Assistant professor – Faculty of computers and information – Beni Suef University

Chapter 2: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

2.3 FTP

2.4 electronic mail
- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P applications

2.7 socket programming with UDP and TCP

Uploading form input

POST method:

- web page often includes form input
- input is uploaded to server in entity body

URL method:

- uses GET method
- input is uploaded in URL field of request line:

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

HTML Forms and Server-side Data

- some web pages allow us to submit our own new data
- most server-side web programs accept **parameters that guide their execution**

Form Example: Creating email account

Form Example: Creating email account

Method types

HTTP/1.0:

- GET
- POST
- HEAD

HTTP/1.1:

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field
- HEAD
  - Used when the client is requesting some information about a resource but not requesting the resource itself.

Method types

HTTP/1.1:

- **CONNECT**
  - Used when the client wants to establish a transparent connection to a remote host, usually to facilitate SSL-encrypted communication (HTTPS) through an HTTP proxy.

HTTP response message

ETags are a mechanism that web servers and browsers use to validate cached components.

status line (protocol status code status phrase)

header lines

```
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 ...
```

Maximum 100 requests in 10 seconds

data, e.g., requested HTML file

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 msg

301 Moved Permanently

- requested object moved, new location specified later in this msg (Location:)

400 Bad Request

- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

User-server state: cookies

many Web sites use cookies

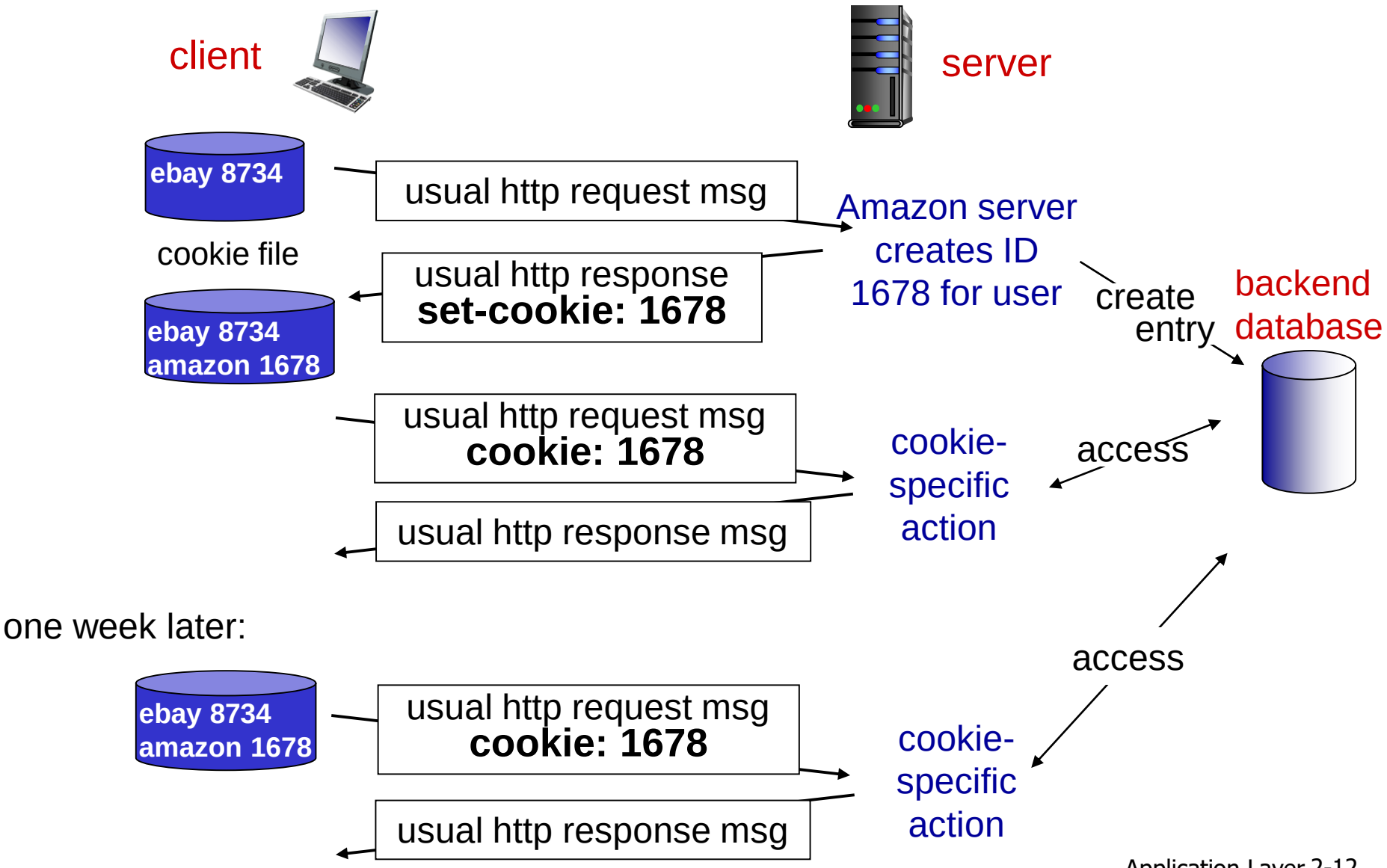
four components:

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:

- Ibrahim always access Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

Cookies: keeping "state" (cont.)



Cookies (continued)

what cookies can be used for:

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

Using Cookies to Store Session Data using ASP.Net(C#)

- Creating a cookie:

```
HttpCookie objCookie = new HttpCookie("myCookie");
DateTime now = DateTime.Now;

objCookie.Values.Add("Time", now.ToString());

objCookie.Expires = now.AddHours(1);

Response.Cookies.Add(objCookie);
```

To create a persistent cookie, specify the expiration time

Retrieving Information from a Cookie using ASP.Net(C#)

- Read the cookie

```
HttpCookie objCookie = Request.Cookies["myCookie"];
```

- Retrieve values from the cookie

```
lblTime.Text = objCookie.Values["Time"];
```

Chapter 2: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

2.3 FTP

2.4 electronic mail
- SMTP, POP3, IMAP

2.5 DNS

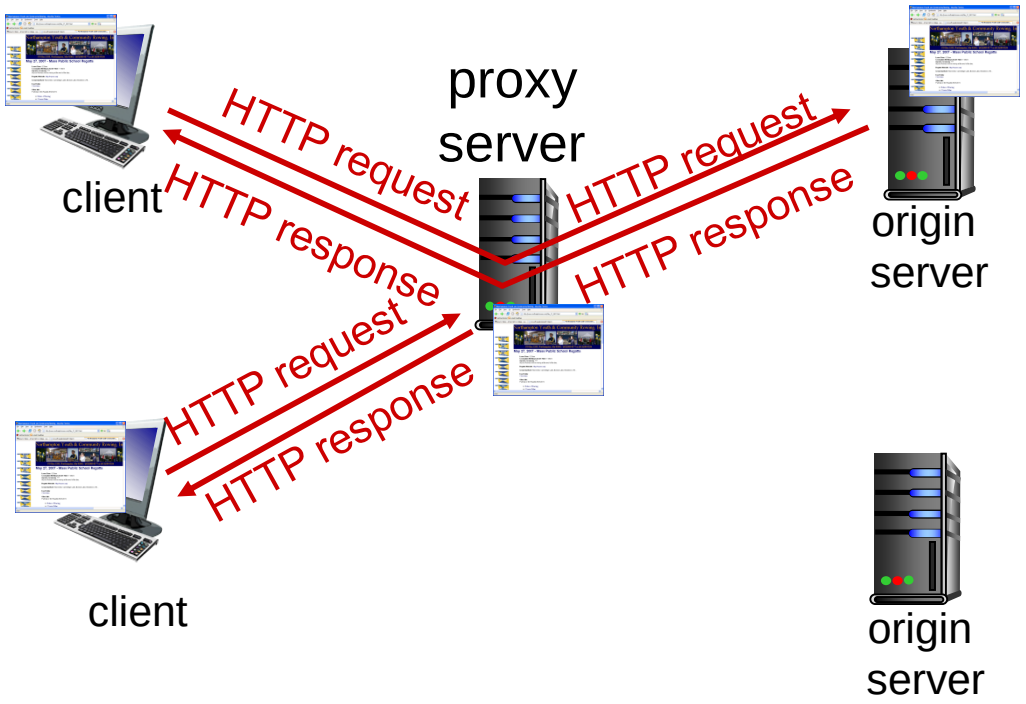
2.6 P2P applications

2.7 socket programming with UDP and TCP

Web caches (proxy server)

goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



What is a Web Proxy Server?

- It is a specialized HTTP Server.
- Functions as a firewall.
  - Protects client computers from Hackers by limiting outside access to clients.
- Allows all clients connected to Web Proxy Server to access Internet from behind “firewall.”

How Does A Web Proxy Server Work?

- Web Proxy Server listens for any request from clients.
- All requests are forwarded to remote internet servers outside firewall.
- Also listens for responses or request from outside the firewall (external servers) and sends to them to internal client computers.
- All that is needed is the proxy server's IP address.

How Does A Web Proxy Server Work?

- Usually, all clients with a subnet use the same proxy server.
- This makes it possible for the proxy server to cache documents that are requested by one or more clients (repeatedly).
- For clients using a web proxy server, it is as if they are getting responses directly from a remote server.
- Caching documents means keeping a copy of internet documents so the server doesn't need to request them over again.

More about Web caching

why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link
- enables "poor" content providers to effectively deliver content (so too does P2P file sharing)

Difference Between a Firewall and a Proxy Server

- A firewall and a proxy server are both components of network security.
- To some extent, they are similar in that they limit or block connections to and from your network, but they accomplish this in different ways.
- Firewalls can block ports and programs that try to gain unauthorized access to your computer, while proxy servers basically hide your internal network from the Internet.

Difference Between a Firewall and a Proxy Server

- Proxy servers works as a firewall in the sense that it blocks your network from being exposed to the Internet by redirecting Web requests when necessary.

- A firewall can prevent programs from running on your computer. A proxy server cannot do this.

Chapter 2: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

2.3 FTP

2.4 electronic mail
- SMTP, POP3, IMAP

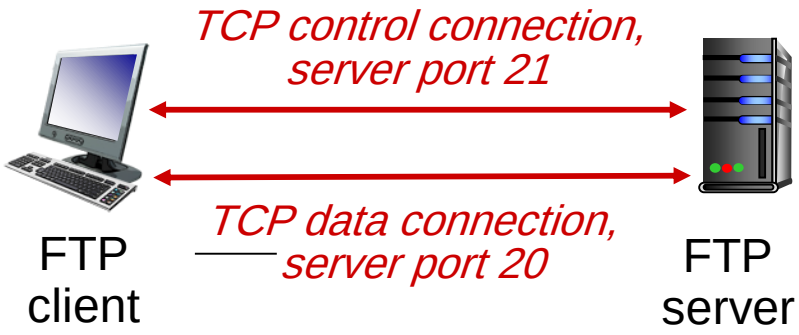
2.5 DNS

2.6 P2P applications

2.7 socket programming with UDP and TCP

FTP: separate control, data connections

- FTP client contacts FTP server at port 21, using TCP
- client authorized over control connection
- client browses remote directory, sends commands over control connection
- when server receives file transfer command, *server* opens *2nd* TCP data connection (for file) *to* client
- after transferring one file, server closes data connection



- server opens another TCP data connection to transfer another file
- FTP server maintains "state": current directory, earlier authentication

FTP commands, responses

sample commands:

- sent as ASCII text over control channel
  - **`USER`** ***`username`***
  - **`PASS`** ***`password`***
  - **`LIST`** return list of file in current directory
  - **`RETR filename`** retrieves (gets) file
  - **`STOR filename`** stores (puts) file onto remote host

sample return codes

- status code and phrase (as in HTTP)
  - **`331 Username OK, password required`**
  - **`125 data connection already open; transfer starting`**
  - **`425 Can't open data connection`**
  - **`452 Error writing file`**

The end