CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

HTTP Message Format

Protocols specify when messages are sent and contents of those messages

Previously described process of sending HTTP messages

Now it's time to find out what they contain

Two types of HTTP messages: requests and responses

Requests -- sent from clients

Responses -- sent from servers

HTTP Request

For example:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

HTTP Request

Message is written in ASCII text

Line endings are carriage return and line feed (`\r\n`). Be careful with this if trying to interact with HTTP servers/clients or writing your own

Request can have as few as one line, but often contains many more to help tweak behavior

HTTP Request

First line is always **request line**:

```
GET /somedir/page.html HTTP/1.1
```

Request line has

- method
- URL
- HTTP version

HTTP Request

URL is self-explanatory

Including HTTP version ensures client and server can communicate correctly and determines what options are available

Method can be one of several values, including:

```
GET, POST, HEAD, PUT, DELETE
```

HTTP Request

Most HTTP requests use `GET` -- request an object

`POST` -- request web page whose contents depend on form data sent by user

`HEAD` -- like `GET`, but server sends HTTP reply without requested object

`PUT` -- upload object to specific path on server

`DELETE` -- delete object on web server

HTTP Request

Not all forms use POST

Many use regular GET but append form data to URL, such as

```
www.somesite.edu/carsearch?chevy&2005
```

HTTP Request

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

HTTP Request

Remaining lines in example are **header lines**

Specify variety of `key: value` pairs

Could be as few as 0 header lines, or could be many more than shown in example

HTTP Request

`Host` -- host on which object resides. Seems redundant (TCP connection with host already exists), but necessary for web proxy caches

`Connection: close` -- do not use persistent connections

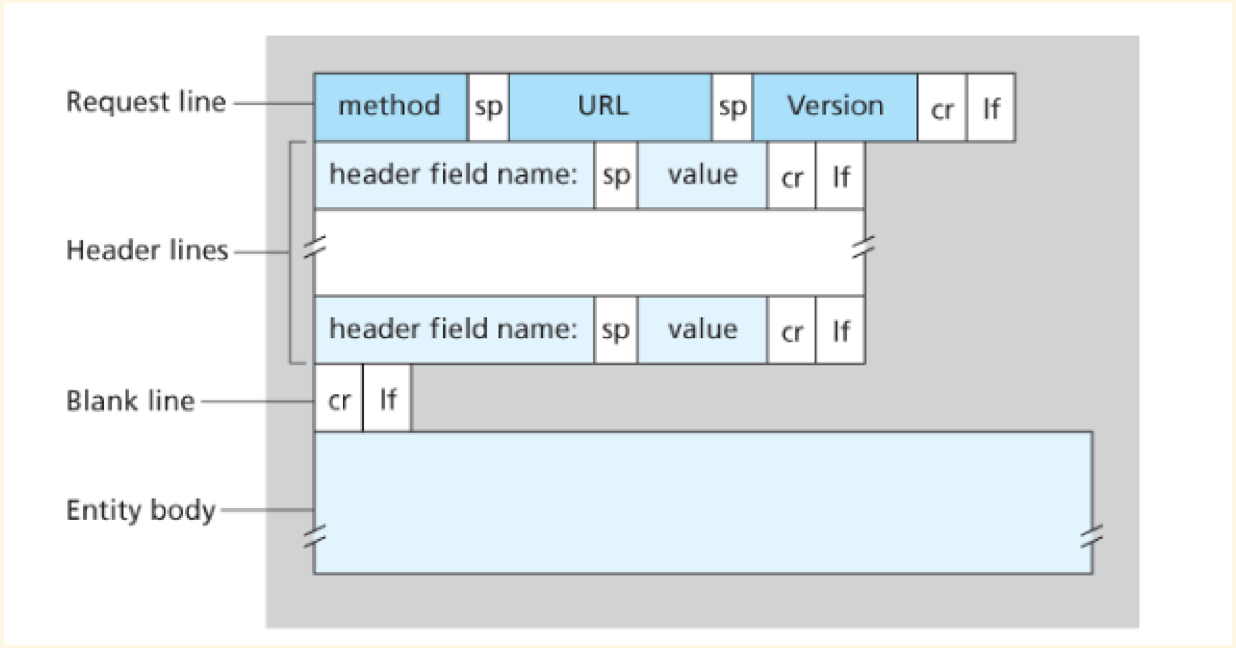
`User-agent` -- specify browser. Server may provide different pages to different browsers for better viewing

...continued...

HTTP Request

`Accept-language: fr` -- browser prefers to be sent French version, if available. Otherwise, send default version

HTTP Request



Next, we examine format of response from server

HTTP Response

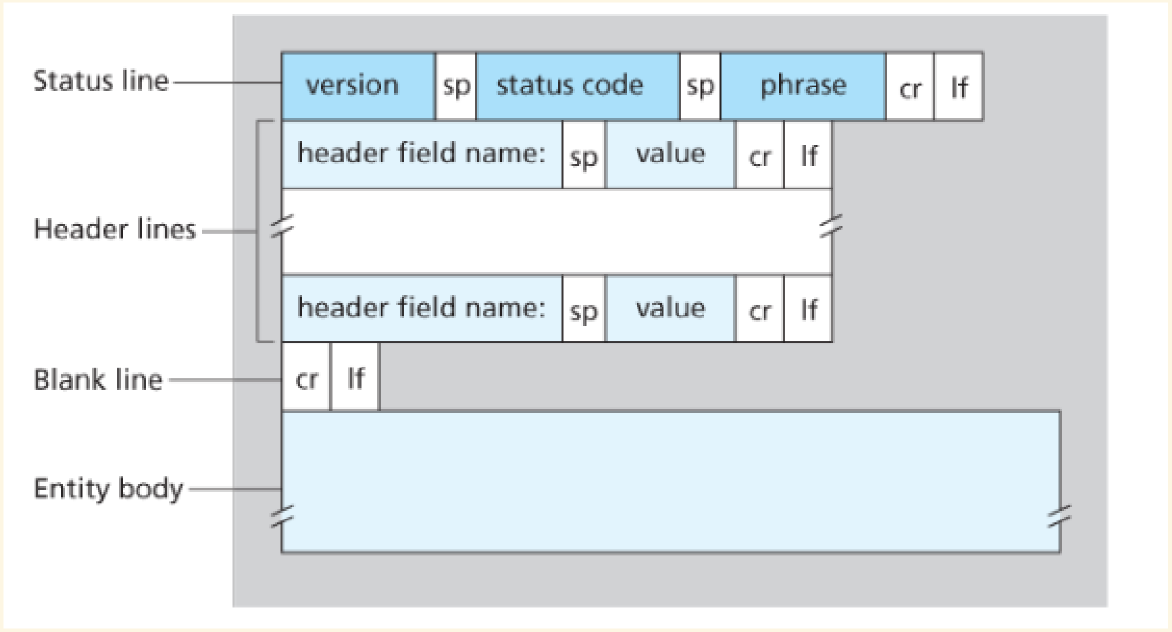
```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data ........)
```

HTTP Response

Consists of

- **status line** (`HTTP/1.1 200 OK`)
- **header lines** (key: value pairs)
- **entity body**

HTTP Response



HTTP Response

Some common status codes and phrases

- `200 OK` -- success
- `301 Moved Permanently` -- requested object moved; new location specified in `location:` header
- `400 Bad Request` -- request not understood (likely bad formatting)

...continued...

HTTP Response

- `404 Not Found` -- requested document does not exist on server
- `505 HTTP Version Not Supported` -- self-explanatory

HTTP Response

Status line holds protocol version, status code, and status message corresponding to status code

`Date` -- date of creation of HTTP response by server, *not* date of creation or modification of requested object

`Last-Modified` -- critical for caching

`Content-Length` -- number of bytes of object being sent

`Content-Type` -- type of object in entity body

HTTP protocol is fairly simple and human-friendly, but powerful enough to form basis of the web