HTTP

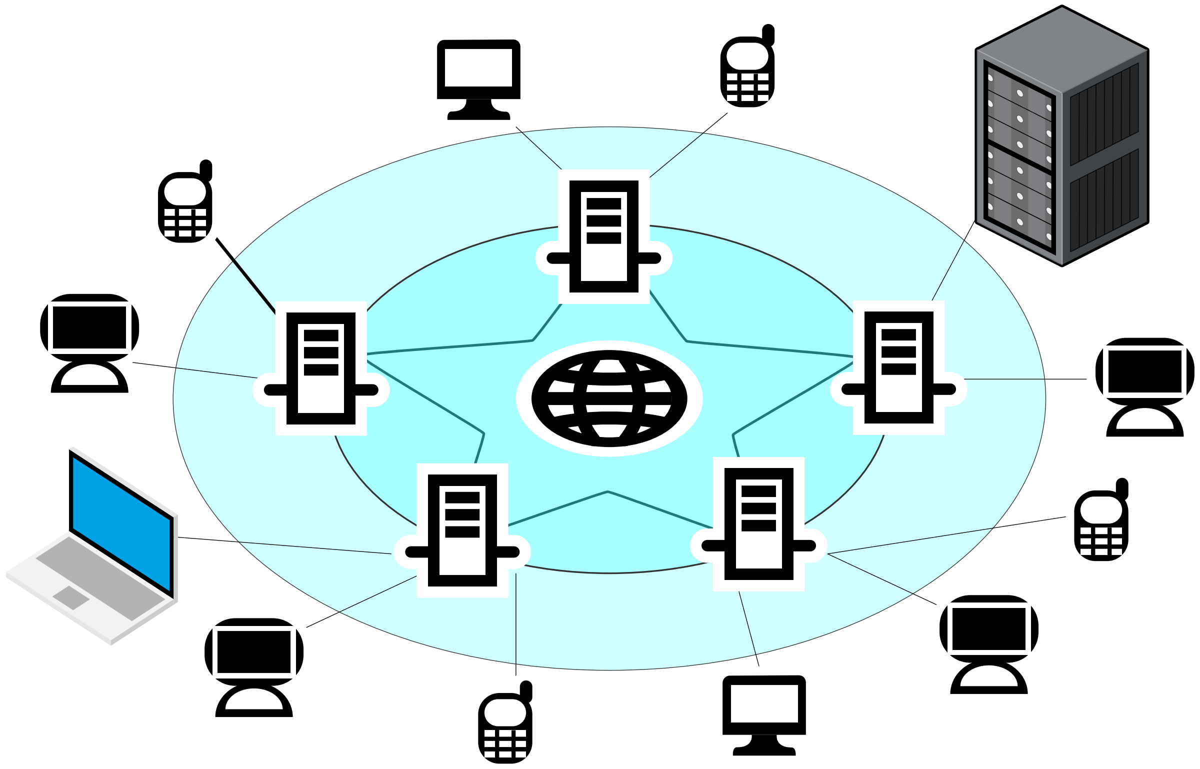
Junghoo Cho

cho@cs.ucla.edu

Web Interaction Example

- http://www.amazon.com

How Does It Work?



Core Internet Standards

- DNS (domain name service)
  - Internet protocol to map domain names to IPs
  - ICANN manages TLD (top-level domains)
- TCP/IP (transmission control protocol and internet protocol)
  - Internet routing and transportation protocol

Core Internet Standards

- HTTP (hypertext transportation protocol)
  - Communication protocol between web servers and web clients
- Encoding standards
  - Text: ASCII, Unicode (UTF-8)
  - Multimedia: JPEG, MP3, H.264, …
- MIME (multipurpose internet mail extensions)
  - Data-type-specification standard

Core Internet Standards

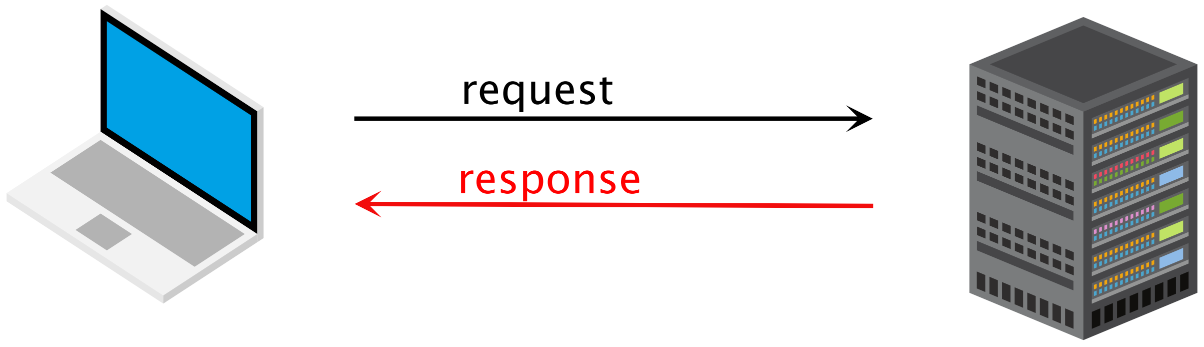
- HTML (hypertext markup language)
  - Markup standard
- CSS (Cascading style sheet)
  - Styling and formatting standard
- Javascript
  - De facto Web programming language

HTTP

- HTTP/2 (2015) is most recent
- HTTP/1.1 (1996) is still extremely popular
  - Major browsers support HTTP/2 only over HTTPS
- Learn HTTP/1.1 first
  - Then talk about HTTP/2
- Two key properties of HTTP
  - Request & response paradigm
  - Stateless protocol

HTTP: Request & Response

- All interactions start with a client's request



HTTP: Stateless Protocol

- Every request is handled independently of others
  - The server is not required to remember “history” of past requests
- Questions
  - Q: What are pros/cons of stateless protocol?
  - Q: what are the implications?

HTTP Example

- Telnet to http://oak.cs.ucla.edu/classes/cs144
- Real HTTP Request by browser

HTTP Message

- A message is either a request or response
- Message structure
  - request/status line
  - header
  - [empty line]
  - body
- Bare minimum HTTP request:

```
GET / HTTP/1.0
```

HTTP Request Example

```
GET /apps/echo.py HTTP/1.1
Host: oak.cs.ucla.edu
Connection: keep-alive
Cache-Control: max-age=0
Dnt: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) Apple
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,ima
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _xsrf=2|11b01aa1
```

HTTP Response Example

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 03:20:33 GMT
Cache-Control: max-age=0, no-cache, s-maxage=10
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 997
Content-Type: text/html; charset=UTF-8
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding

<html>
<head><title>Example page</title></head>
...
```

HTTP Request Line

- `METHOD PATH PROTOCOL_VERSION`

```
GET /apps/echo.py HTTP/1.1
```

HTTP Methods (1)

- `GET`: “retrieve” data
  - **IMPORTANT:** NO significant side effect at server
- `POST`: “post” data at the specified URL
  - May leave a side-effect on the server
- `PUT`: “place” the data at the URL (~ replace the data)
- `DELETE`: “delete” the data at the URL

HTTP Methods (2)

- Less common:
  - `HEAD`: same as `GET` but requests header only
  - `OPTIONS`: requests information on options available at the server
  - `TRACE`: the final recipient returns the whole request message in the r body

HTTP Header

```
GET /apps/echo.py HTTP/1.1
Host: oak.cs.ucla.edu
Connection: keep-alive
Cache-Control: max-age=0
Dnt: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) Apple
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,ima
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _xsrf=2|11b01aa1
```

HTTP Response

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 03:20:33 GMT
Cache-Control: max-age=0, no-cache, s-maxage=10
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 997
Content-Type: text/html; charset=UTF-8
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding

<html>
<head><title>Example page</title></head>
...
```

HTTP Response

- Structure
  - Status line
  - Header
  - [empty line]
  - body
- Status line
  - `VERSION STATUS_CODE REASON_PHRASE`
  - Example: `HTTP/1.1 200 OK`

Status Code

- 2xx: Success
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

HTTP Response Header

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 03:20:33 GMT
Cache-Control: max-age=0, no-cache
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 997
Content-Type: text/html; charset=UTF-8
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding

<html>
<head><title>Example page</title></head>
...
```

Questions?

HTTP/2: Background

- Many high-latency limited-bandwidth mobile devices
- Many objects are needed to display a single page
  - HTML, image, CSS, JavaScript, …
  - ~ 100 objects, ~ 2MB

HTTP/2

- Key assumptions have been relaxed for speed and efficiency
  - Server push
  - Stateful header compression
  - Binary encoding
  - …

HTTP/2 New Features (1)

- “Multiplexed streams”
  - Multiple outstanding requests through a single connection
  - Split messages into small frames
  - Priority specification
- HPACK
  - Stateful HTTP header compression

HTTP/2 New Features (2)

- “Server push”
  - Predictively cache pushed by the server
- Binary format
- More detail at https://daniel.haxx.se/http2/

References

- HTTP/1.1: RFC 7230 – RFC 7237
- HTTP/2: RFC 7540