CISC 3120

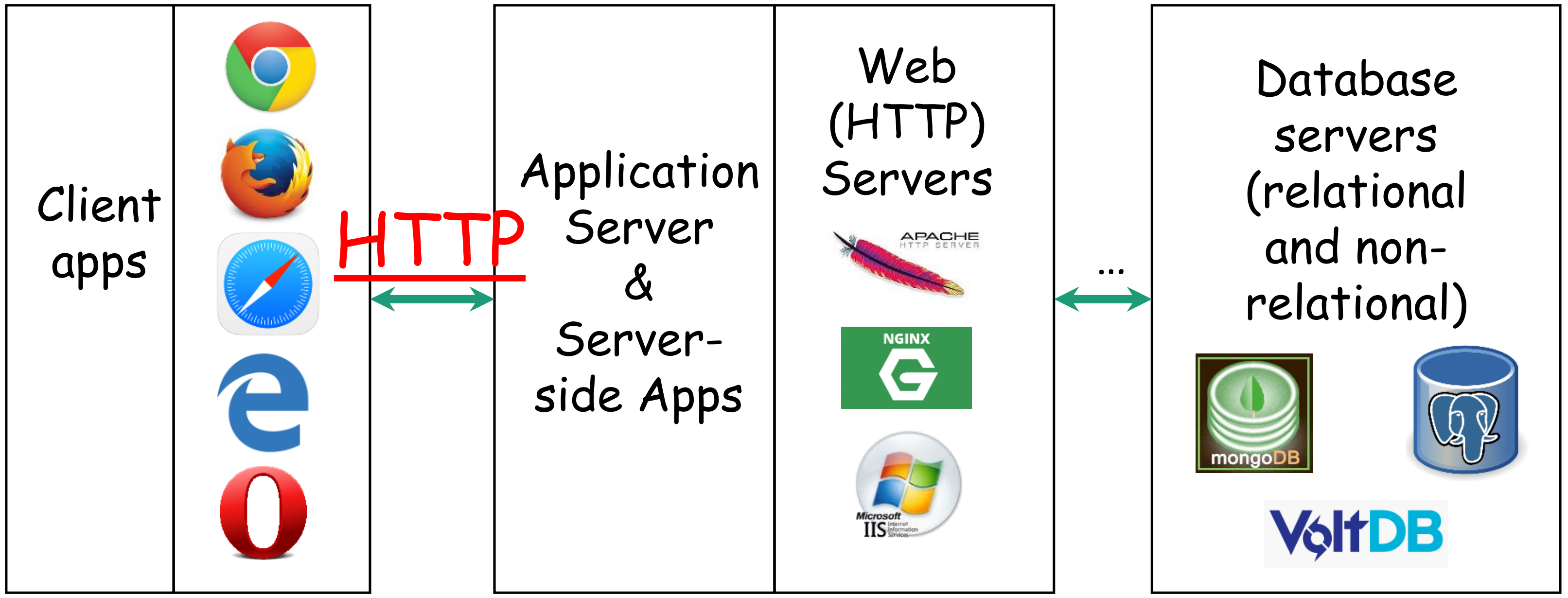
C22: Browser & Web Server Communication

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Web Application Architecture


Client apps
HTTP
Application Server & Server-side Apps
Web (HTTP) Servers
APACHE HTTP SERVER
NGINX
Microsoft IIS
...
Database servers (relational and non-relational)
mongoDB
VoltDB

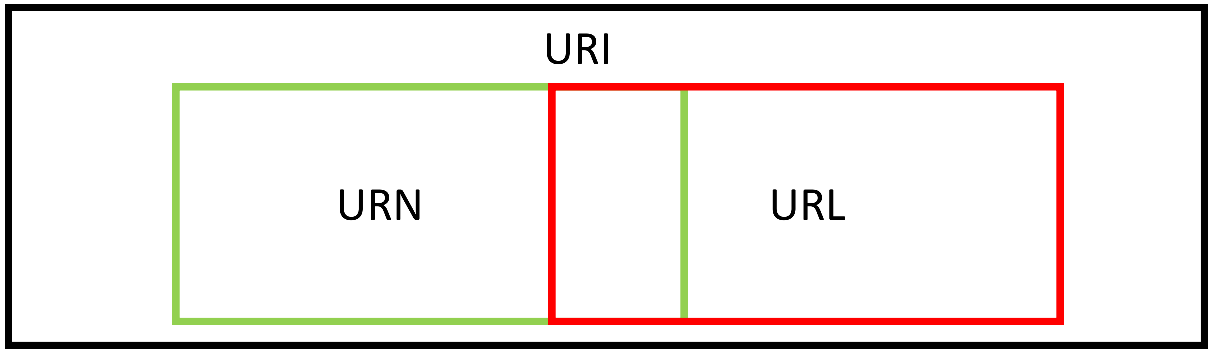
Web Client

Web and Application Servers

Storage System (Database Server)

URI, URL, and URN

- Defined in
  - RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax (obsoletes RFCs 2396, 2732)
- Updated by
  - RFC 6874 and RFC 7320.



URI

- Uniform Resource Identifier
  - A means for identify a resource
  - A sequence of characters from a very limited set
    - The basic Latin alphabet, digits, and a few special characters

URI: Examples

- These are examples of URIs

  ftp://ftp.is.co.za/rfc/rfc1808.txt

  http://www.ietf.org/rfc/rfc2396.txt

  ldap://[2001:db8::7]/c=GB?objectClass?one

  mailto:John.Doe@example.com

  news:comp.infosystems.www.servers.unix

  tel:+1-816-555-1212

  telnet://192.0.2.16:80/

  urn:oasis:names:specification:docbook:dtd:xml:4.1.2

URI: Syntax

- Syntax
  - URI = scheme ":" hier-part ["?" query] ["#" fragment]

```
foo://example.com:8042/over/there?name=ferret#nose
\_/   \______________/\_________/ \_________/ \__/
 |           |            |            |        |
scheme     authority       path        query   fragment
 |   _____________________|__
/ \ /                        \
urn:example:animal:ferret:nose
```

URL

- Universal Resource Locator
  - A subset of URIs
  - Identity the resource
  - Locate the resource
    - by describing its primary access mechanism in the URI (e.g., its network "location").

Locate Resource/Object on the Web with URL

- Example:

  http://www.sci.brooklyn.cuny.edu/course/CISC3120/lecture/cisc3120_c21.pdf#page=3

  - Access mechanism: network protocol HTTP
    - Protocol: HTTP
    - Port: 80
    - Hostname: www.sci.brooklyn.cuny.edu
    - Name of the resource: /course/CISC3120/lecure/cisc3120_c21.pdf
    - (optional) Query: none for this example
    - (optional) Name fragment: #page=3

More Discussion on URI and URL

- See Java API
  - java.net.URI and java.net.URL
- Convert URI to URL
  - URI has a method called toURL
    - public URL toURL() throws MalformedURLException

URI Decode and Encode

- Decode and encode
  - RFC 2396 defines an “escaping” scheme (e.g., what if the name has space, “:”, or “/” etc
- URI’s toURL method does encoding
  - http://foo.com/hello world/ → http://foo.com/hello%20world
- Two classes
  - java.net.URLDecoder and java.net.URLEncoder

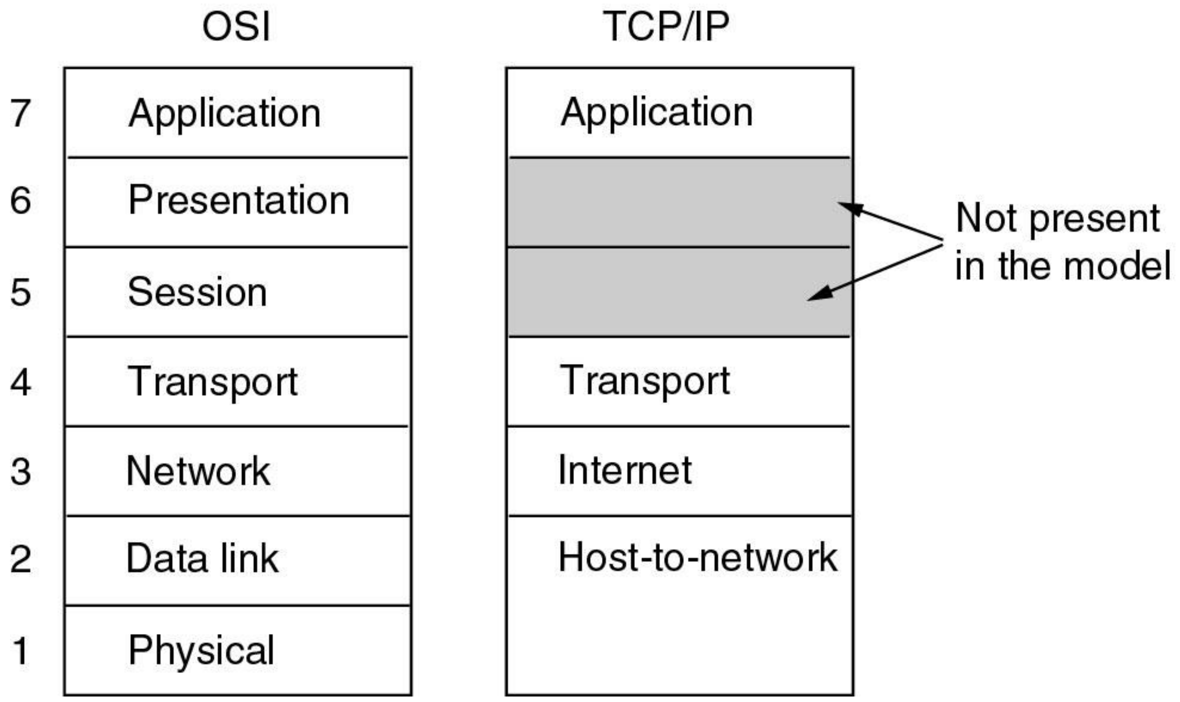
Example: Locate a Resource on the Web

- See URLReader in the “network” directory of the “sampleprograms” repository
- What do you observe?

HTTP

- Hypertext Transfer Protocol
  - Simple request-response protocol layered on TCP/IP
    - Where does it belong in the OSI 7-layer model and the TCP/IP model?

HTTP: An Application Layer Protocol



HTTP Message Exchange

- A typical scene involves a request and response cycle
    - A client establishes a connection to the sever
    - The client sends a HTTP request along the connection to the server
    - The server replies the client with a response
    - The client reads from the connection the response from the web serve

Example: HTTP/1.1 Request

- Header and Body

header

```
GET /index.html HTTP/1.1
User-Agent: Java Web Client
Host: localhost:61235
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive


```

Request method, URN, protocol version

body

Two blank lines

HTTP Request Methods

- GET
    - fetch a URL
- HEAD
    - fetch information about a URL
- PUT
    - store to an URL
- POST
    - send form data to a URL and get a response back
- DELETE
    - delete a URL
- Most frequently used methods are GET and POST

Example: HTTP/1.1 Response

- Header and body

```
HTTP/1.1 200 OK

Date: Thu, 16 Nov 2017 22:06:47 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips mod_fcgid/2.3.9
PHP/5.4.16 mod_wsgi/3.4 Python/2.7.5
Last-Modified: Tue, 11 Jan 2000 21:02:46 GMT
ETag: "17b-35ddc09a30980"
Accept-Ranges: bytes
Content-Length: 379
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<HTML>
…
</HTML>
```

protocol version, status code, status message

header

Blank line

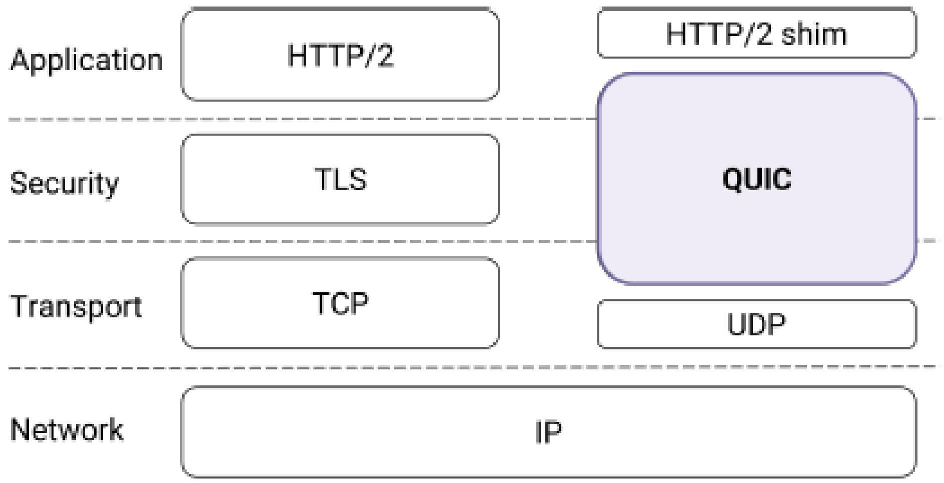
body

HTTP Evolution

- HTTP 0.9 – 1.0: initial development; 1991 – 1996
  - Allows only one outstanding request at a time on a given TCP connection
- HTTP/1.1: standardized in 1997
- HTTP/2: standardized in 2015
  - Aimed reduce latency
  - Allows interleaving requests & responses on the same connection, reduces header size, supports prioritization of requests

Some Recent Development

- Web becomes an application platform
- Secure Web traffic becomes dominant
- Handshake latencies
    - TCP: 1 round-trip delay; TLS: 2 round-trip delay



Langley et al., 2017

Question?

- How web browser and Web server communication?
- How do we locate a resource on the Web?
- Evolving to HTTP/2