Chapter 1: An introduction to web engineering

COURSE : WEB ENGINEERING WITH LAB

COURSE CODE: CIS 331 & 331L

COURSE LEADER: ABDULLAH BIN KASEM BHUIYAN

LECTURER, DEPT. OF CIS

DAFFODIL INTERNATIONAL UNIVERSITY

EMAIL: ABDULLAH16-329@DIU.EDU.BD

MOBILE: 01831661534

Learning outcomes

- Describe how clients and web servers interact
- Request resources from servers and understand their response
- Describe different URL components
- Understand and use different HTTP Headers
- What is web engineering

Historical development

1945	*article by Vannevar Bush* in "Atlantic Monthly": proposal of a photo-electrical mechanical device called a *Memex* (memory extension) which could make and follow *links* between documents on microfiche
1965	*article by Ted Nelson "A File Structure for the Complex, the Changing, and the Indeterminate"* first mention of the term "*Hypertext*"
1968	*NLS (oNLine System) by Engelbart* first implementation of a *hypertext system*
1969	*ARPANET* the world's first operational packet switching network and the progenitor of the Internet

Historical development 1

1974	*article "A protocol for Packet Network Interconnection"* introduction of *TCP (Transfer Control Protocol)*
1978	*IP (Internet Protocol)*
1984	*Domain Name System (DNS)*
1989	*"Information Management: A Proposal" by T. Berners-Lee* "hour of birth of the WWW"

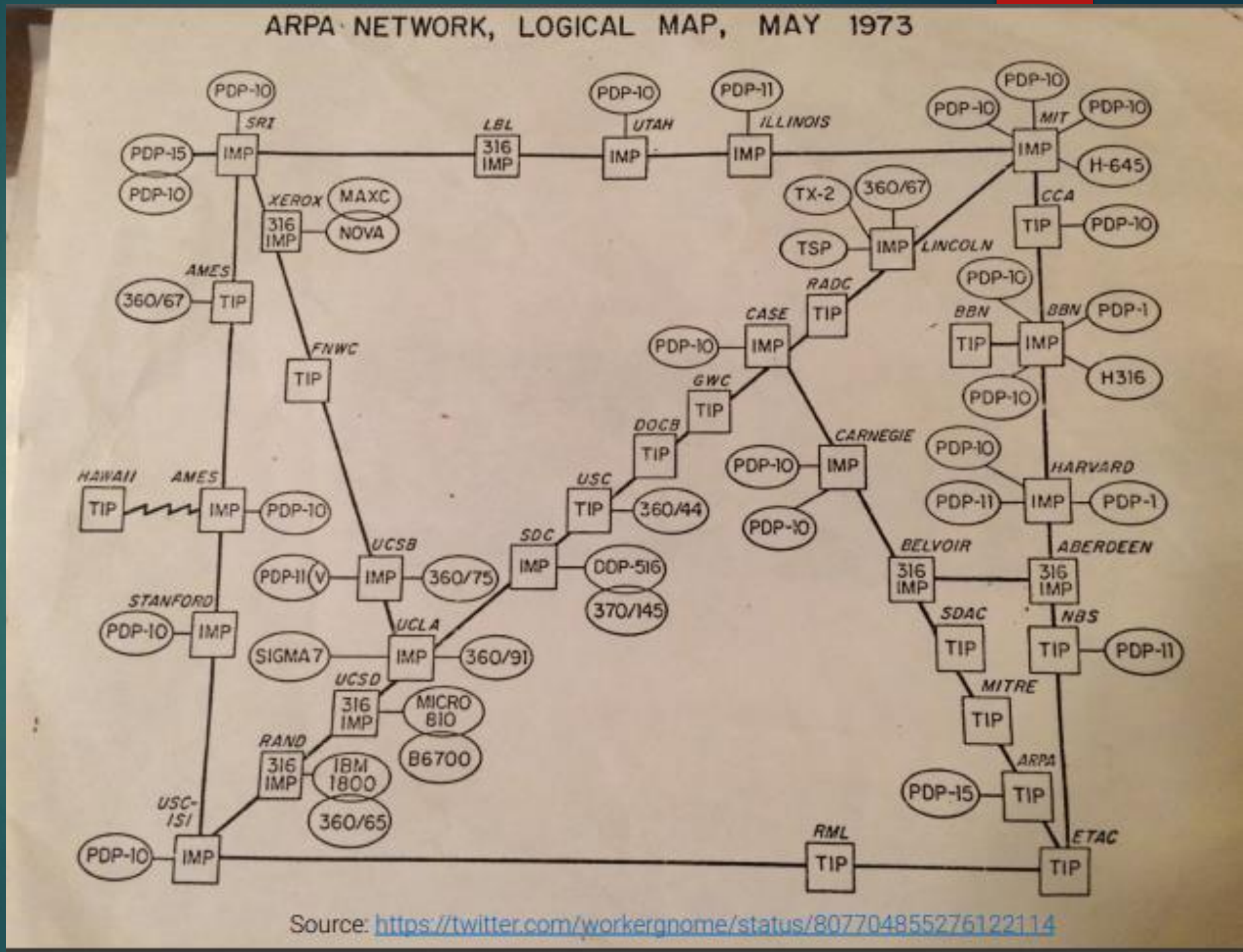
Historical development 2

1990	*First command-line browser*
1993	*Release of 1st graphical web browser: Mosaic*
1994	*Internet access by dial-up systems (like CompuServ, AOL)* *Foundation of the W3C* *Netscape Navigator 1.0*
1998	*Google is founded in Menlo Park, California* *"The PageRank Citation Ranking: Bringing Order to the Web" by L Page, S Brin, R Motwani, T Winograd (Stanford)*

What is Internet?

- The internet is the global system of interconnected computer networks that uses the Internet protocol suite (TCP/IP) to link devices worldwide.
- Important concepts
  - TCP (Transmission Control Protocol) — connection oriented protocol Establishes a point-to-point connection between two entities in the network
  - IP (Internet Protocol) — principal communications protocol on the internet Delivers packets of data across network boundaries
  - IP Address — numerical label assigned to devices in a network that use the internet protocol to communicate with other devices

State of the "Internet" (ARPANET) in 1973



Source: https://twitter.com/workergnome/status/807704855276122114

High Level Web Overview

- What happens if we request a website from the internet?

- What are the steps executed in the background required to display a website

High Level Web Overview

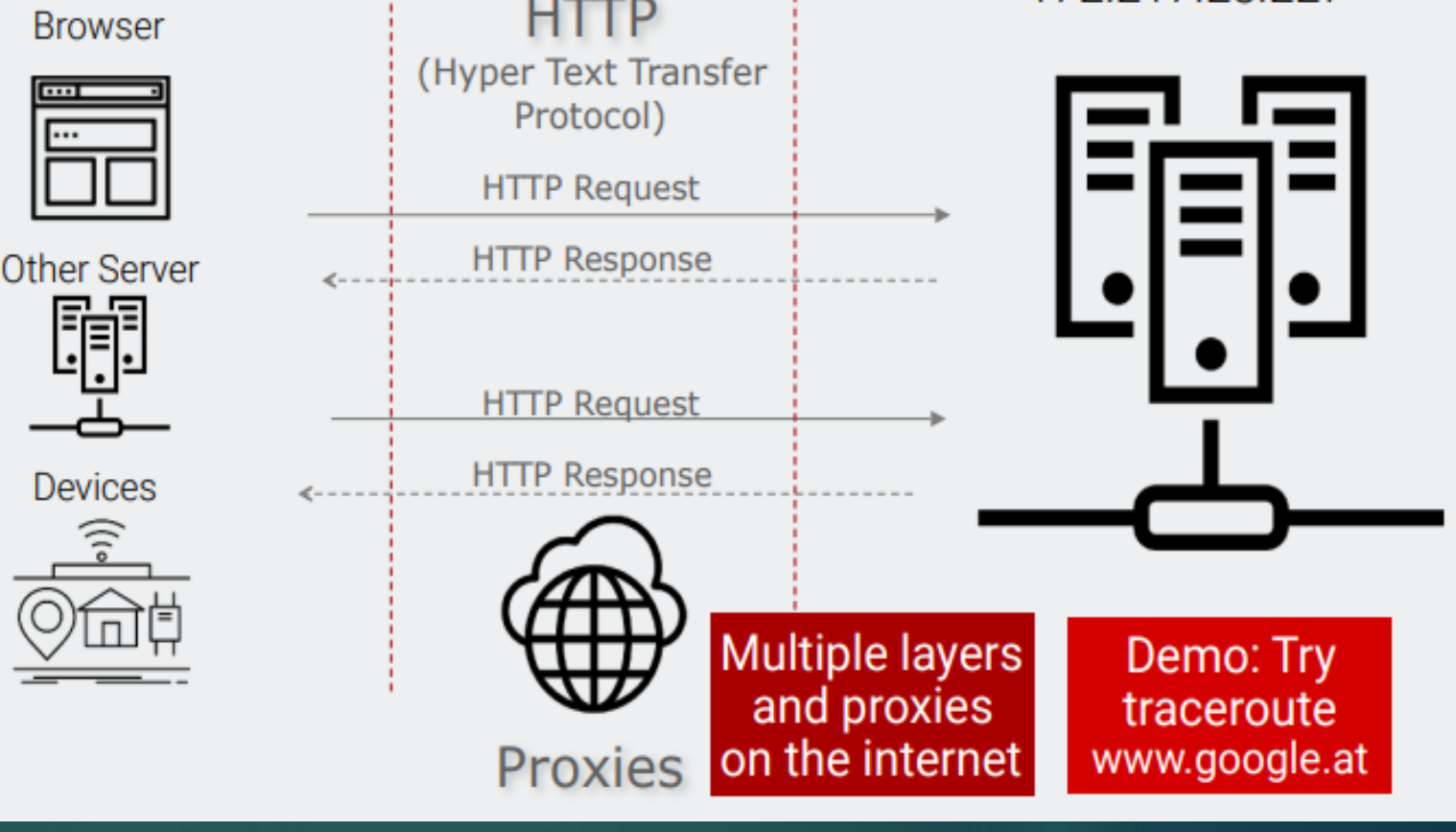
- Servers wait for requests
- They serve web **resources**

Client

https://www.google.at

www.google.at –> 172.217.23.227

Domain Name System (DNS): Translating hostname to IP address

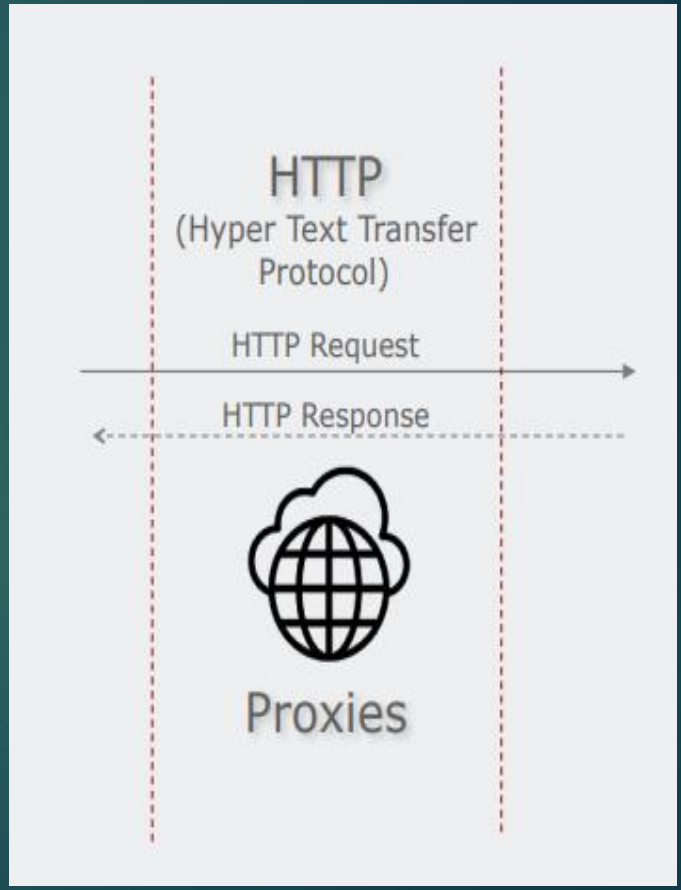


Multiple layers and proxies on the internet

Demo: Try traceroute www.google.at

HTTP Overview

- Builds upon TCP/IP
- Synchronous request-response protocol
  - Client (web browser) sends request
  - Web server replies with appropriate answer (could also be an error)
- "Stateless" protocol
  - Each request-response pair is independent
  - No permanent connection between server and browser (allows for a high number of users per server)
- Proxies mediate between browser and server (caching, filtering, etc.) •
- In HTTP everything is sent and received as clear text
  - Use HTTPS: HTTP over a secured (TLS) connection



HTTP Resources and URLs

Resource

- Abstract concept for nodes in hypertext
  HTML files, documents, images, etc.
- Data types defined by MIME (RFC 2045)
  "text/html", "image/png", “application/xml“, etc.

Uniform Resource Locator (URL)

- Standardized way of identification and addressing of any resource on the internet
- Subtype of Uniform Resource Identifier (URI)

URL Syntax

<scheme>://[<user>[:<password>]@]<server>[:<port>]/[<path>][?<query>][#<fragment>]

HTTP Resources and URLs - Syntax

<scheme>://[<user>[:<password>]@]<server>[:<port>]/[<path>][?<query>][#<fragment>]

https://usr:pwd@tennis-club-wieden.at:3000/members/rackets?year=2020#vintage

Scheme

- Protocol to be used when connecting to a server http(s), ftp, mongodb, etc.

User/Password

- Optional: Credentials to access a protected resource

Server

- Domain name or IP address of the server

https://tiss.tuwien.ac.at/education/course/courseRegistration.xhtml?courseNr=188951&semester=2020S

Port

- Port at which the server is listening for requests

Path

- Local path to a resource on the server

Query

- Parameters that can be passed to server app

Fragment

- Name of an entity within the resource. **This is only used by clients**

HTTP Request

- Refers to a certain resource (identified by its URL)
- Contains a certain type ("method")

  Most common methods for access: GET, POST, PUT
- Can contain application data ("body"), e.g., the data of a form (POST, PUT)
- Can contain application metadata, e.g.:
  - Preferred data type and language (for GET, POST) – Content Negotiation
  - Data type of the body (for POST, PUT)
- Can contain request metadata (headers)
  - Target host, User authentication, Cookies, etc.

Which resource are we retrieving

How are we retrieving a resource

What data/payload are we sending to the resource

What **data type** do we want from the resource (HTML, JSON)

What **kind of data** are we sending

HTTP Request Method

- Each access to a resource has a certain request type ("method")
- **GET**: request a resource, only retrieves data
- **POST**: submit data to a resource
  - Data is included in body of the request
  - May result in creation of new resource or update of existing resource
- **PUT**: replaces target resource with sent payload
- **DELETE**: delete a resource
- **PATCH**: provides a set of instructions to modify the target resource
- **OPTIONS, TRACE, HEAD, CONNECT:** access to the metadata of the servers, the Internet connection, the resource, etc.

Safe and repeatable
(expect no side effects)

Expect Side Effects
for POST, PUT, DELETE, PATCH

Idempotent
(expect same effect even with multiple executions)

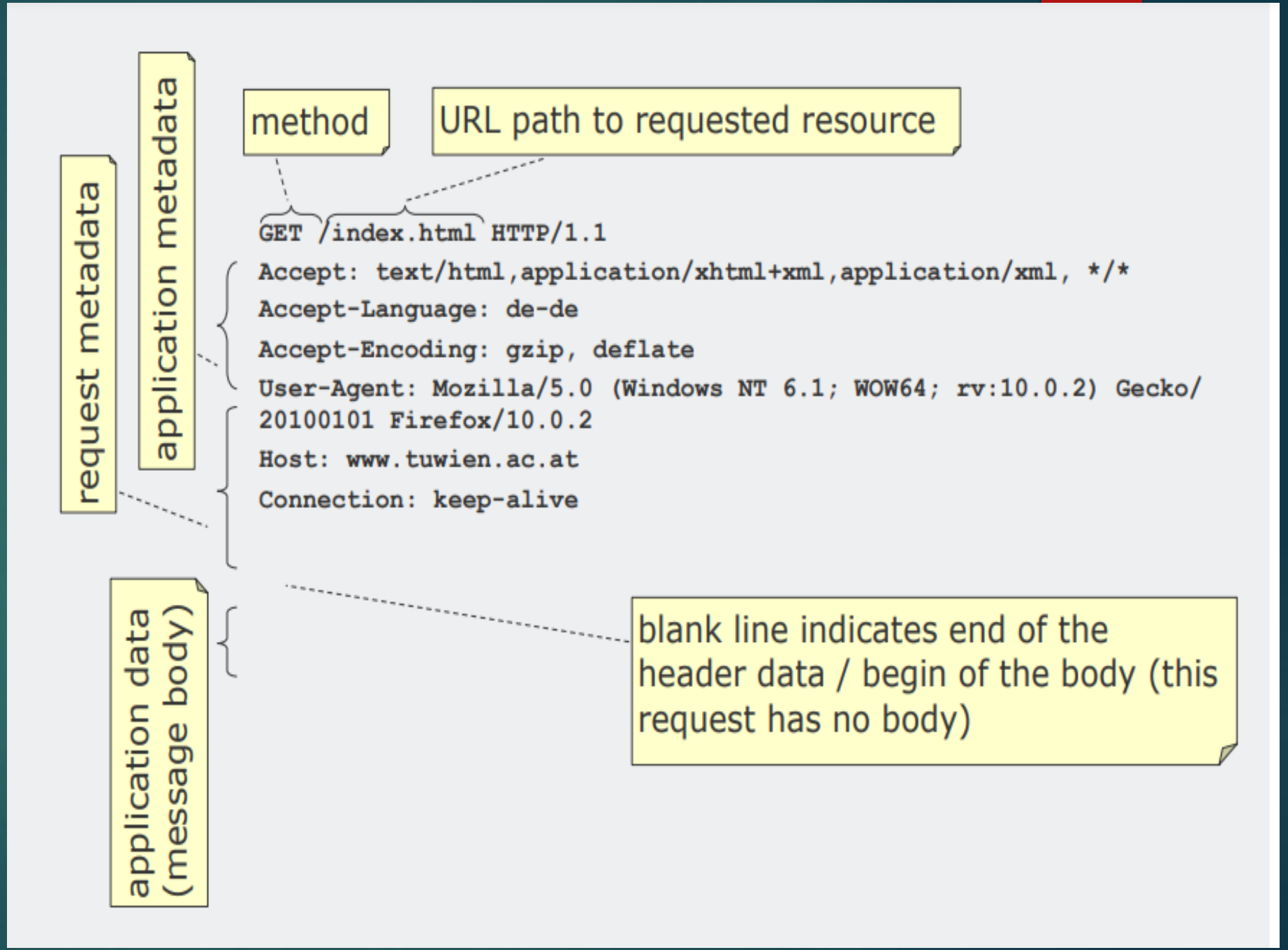
HTTP Request Headers - Examples

- **Accept**: what kind of response type to accept
  - Accept: application/json
- **Content-Type:** what kind of request payload are we sending (in POST and PUT)
- **Accept-Encoding:** tells server a list of acceptable encodings
  - Accept-Encoding: gzip, deflate
- **Authorization**: Authorization method and credentials
  - Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l
- **Cookie**: Sends a cookie to the server (more on that later)

MIME Types

text/plain
text/html
image/jpeg
application/pdf
application/xml

HTTP Request Example



HTTP Response

- Always follows a request message C
- Contains a status code
- Can contain application data („body“)
- Can contain application metadata, e.g.:
  - Data type and encoding of the application data
  - Caching possibilities and expiring date
  - Current URL of a transferred resource (for GET)
- Can contain response metadata, e.g.:
  - Server, TCP connection state, date

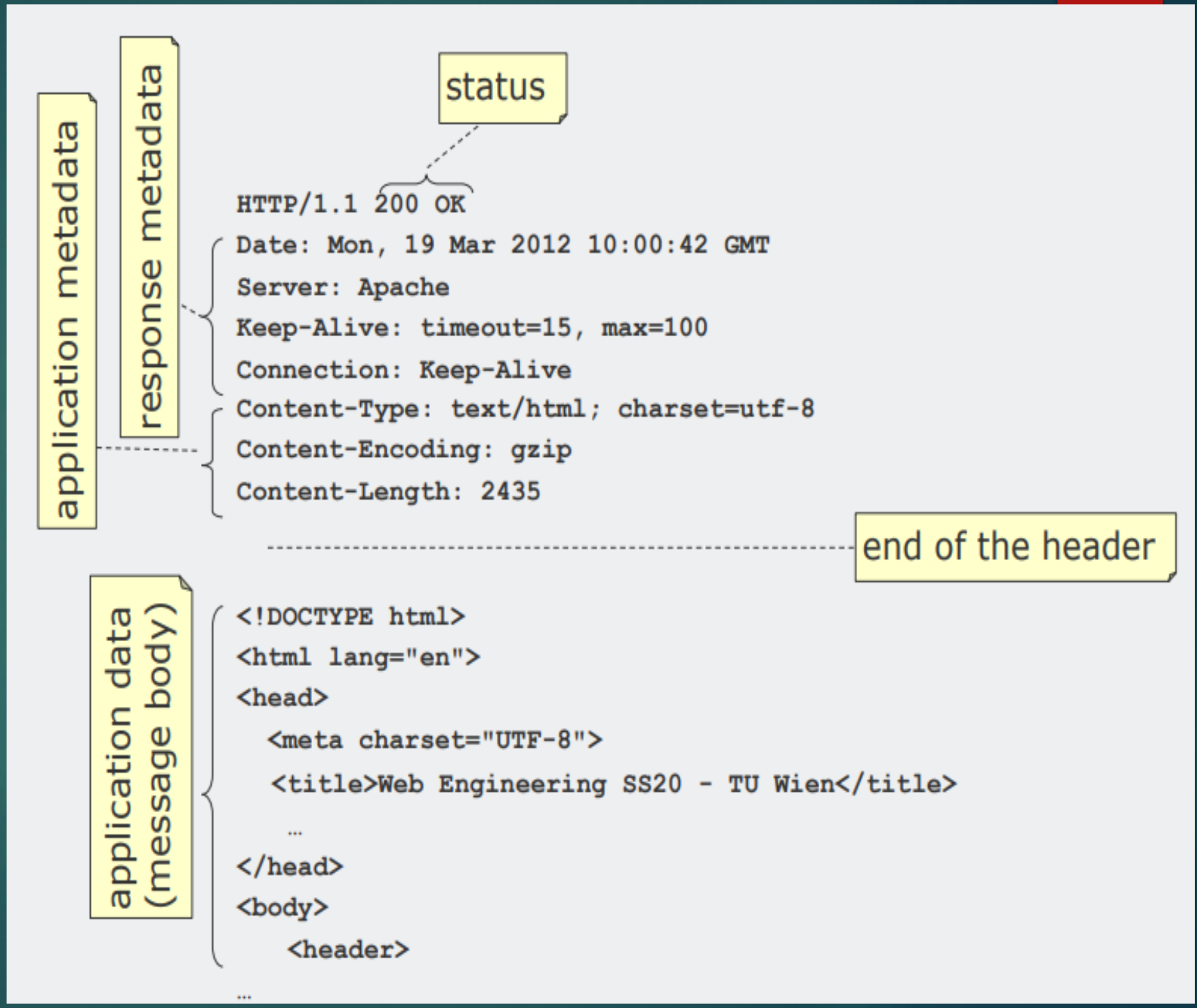
Status

Code	Description	Common Example
1xx	Informational	101 Switching
2xx	**Success**	**200 OK**
3xx	Redirected	301 Permanent
4xx	**Client Error**	**404 Not Found**
5xx	**Server Error**	**500 Internal Server Error**

HTTP Response Headers - Examples

- Expires: time/date the response is considered "stale" (used for caching)
  - Expires: Wed, 21 Oct 2020 07:30:00 GMT
- Last-modified: contains the date the resource was modified
- Content-Type: media type of the resource
  - Content-Type: text/html; charset=UTF-8
- Set-Cookie: saves a cookie on the client side (more on that later)

HTTP Response Example



HTTP Live Demos - Summary

- A variety of HTTP requests with curl:
  - Retrieving textual data and displaying
    - Response headers in (1),
    - and verbose output (TCP information, request and response headers (2)

    Overall goal here is to show the different content types

    (1) curl --head http://people.csail.mit.edu/jcito/we/some_text.json

    (2) curl -v http://people.csail.mit.edu/jcito/we/some_text.txt
- Create a request bin to display - I am replacing the actual URL with $URL here

  (this can be also done in the command line by saying export URL= https://enkuj0njhbzm.x.pipedream.net/)

  -v = verbose, -H sets requests headers, -d sets request body, -X sets the request method
  - Show GET: curl -v -X GET $URL
  - Show POST: curl -v -d '{ "name": "Jurgen", lastname: "Cito" }' -H "Content-Type: application/json" $URL
  - Show PUT with custom header: curl -X PUT -H "Authorization: Basic XYZ" $URL
- Go to Chrome and open More Tools -> Developer Tools, select tab "Network" (check "Disable Cache")
  - Go to website of your choosing and see a horde of HTTP requests coming in

What is Web Engineering?

- Web Engineering is the application of systematic and quantifiable approaches (concepts methods, techniques tools) to cos t - effective requirements analysis, design, implementation , testing, operation , and maintenance of high -quality Web applications

The top problem areas of large -scale Web application projects

- Failure to meet business needs (84%)
- Project schedule delays (79%)
- Budget overrun (63%)
- Lack of functionalities(53%)
- Poor quality of deliverables (52%)

Reference book

- 1. “**Web Site Engineering: Beyond Web Page Design” By** Thomas A. Powell, David L. Jones and Dominique C. Cutts
- 2. “**Web Engineering: The Discipline of Systematic Development of Web Applications” By Gerti Kappel, Birgit Prýýll, Siegfried Reich and Werner Retschitzegger**