Entwurf Verteilter Systeme

Prof. Dr.-Ing. Martin Gaedke

Technische Universität Chemnitz

Fakultät für Informatik

Professur Verteilte und selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

Section

HYPERTEXT TRANSFER PROTOCOL (HTTP)

Hypertext Transfer Protocol

- **Hypertext Transfer Protocol (HTTP)** is used to exchange resources (such as websites, pictures, JavaScript, other MIME-types resources) between a user agent and a server following the Request/Response model.
  - Basic web resource transfer mechanism
  - IETF-Standard: RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1
  - Development: World Wide Web Consortium (W3C)
  - Is a transfer protocol, not a transport one
- Protocol properties
  - Exchange of Request/Response data based on TCP/IP
  - Stateless communication between user agent and server
  - Two message types: Request and Response
  - Messages are ASCII-encoded
  - Messages are used to realize methods: GET, POST, HEAD, etc.

HTTP Versions

- Early onset - Version 0.9
  - Only one command is supported: "GET" (no other approaches, like, for example, attributes)
  - Is not extensible, no versioning support
- Ideas of versioning and robustness of the Web
  - Use of version numbers for HTTP (RFC 2145, May 1997)
  - Robustness principle (backward compatibility support)
- May 1996 - Version HTTP/1.0 (RFC 1945)
  - Introduction of version numbers
  - Stateless protocol (TCP-Slow-Start problematics)
  - Support for extensions
  - HTTP/1.0 is still widespread
- June 1999 Version HTTP/1.1 (RFC 2616), supplemented by TLS (RFC 2817)
  - Enables persistent connections and proxy use
  - Many improvements in terms of performance, i.e. Request Pipelining or Multihomed Server
  - Addition of a secure connection possibility by means of Transport Layer Security (TLS)

Generic Approach

- Generic message structure enables extensions
- Important: The concept in question is used in all protocols that build upon HTTP

- **Generic-Message = Start-Line**
- ***Header**
- **CRLF**
- **[Message-Body]**

- *Start-Line ::=* Request-Line | Response-Line

- *Header ::=* field-name ":" [field-value] CRLF
  - field-name = token
  - field-value = *(field-content | LWS)
  - LWS = Linear White Space

- *Message-Body*
  - If exists MUST be encoded
  - Presence signaled by header field Content-Length or Transfer-Encoding

HTTP-Request Message

- Message structure

  <Method>“ ”<URI>“ ”<Protocol>

  <Headers>

  CRLF

  [<Data>]

- *Method* ::= “GET” | “POST” | “HEAD” | …
- *Protocol* ::= “HTTP/1.0” | “HTTP/1.1” | …
- *Headers* ::= <hName>“:”<hValue>
  - hName – header name h (Attribut-Name)
  - hValue – Value of the value space of header h (Attribut-Wert)
- *Data* ::= <TEXT>

HTTP-Response Message

- Message structure

  <Protocol>“ ”<Status-Code>“ ”<Reason-Phrase>

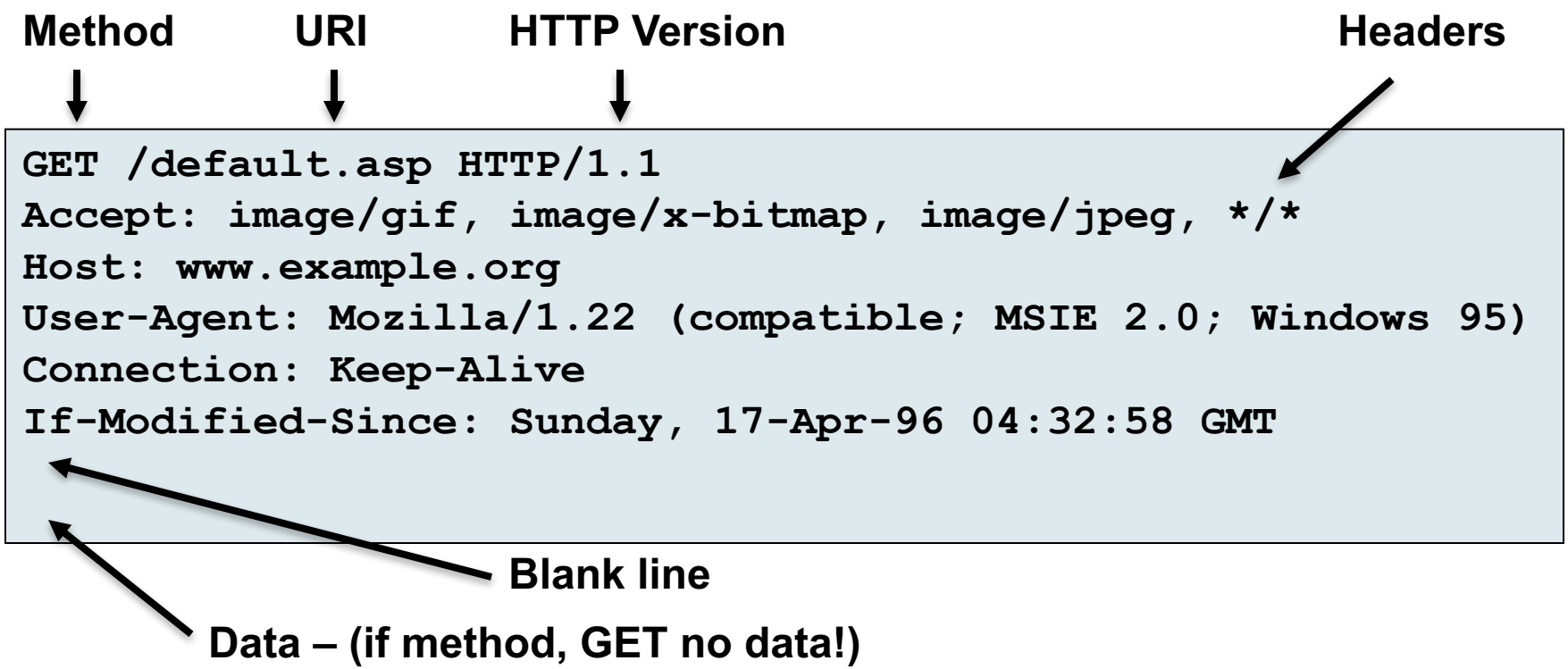
  <Headers>

  CRLF

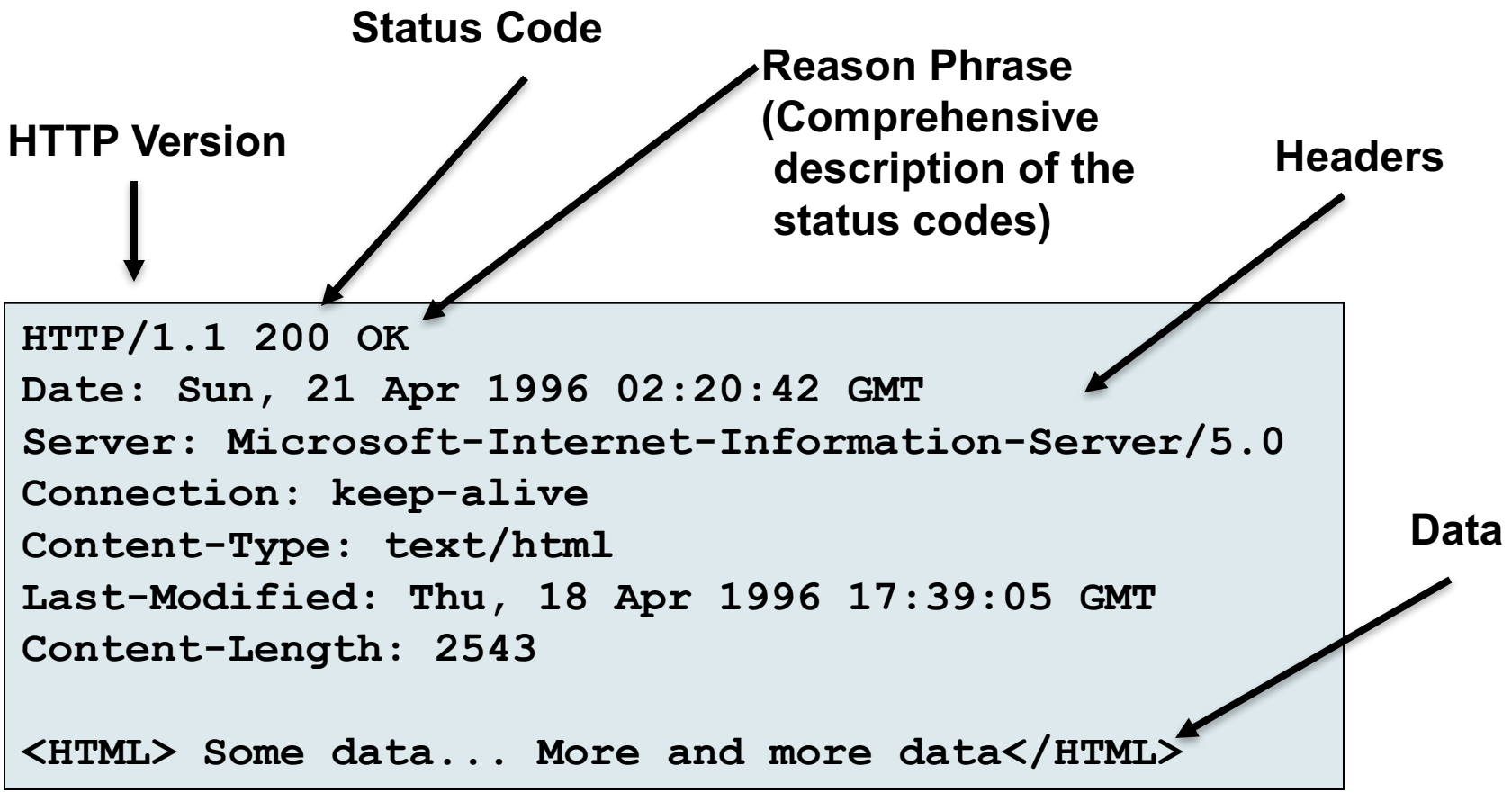
  [<Data>]

- Protocol ::= “HTTP/1.0” | “HTTP/1.1”
- Status-Code ::= DIGIT+ ; for use by automata
- Reason-Phrase ::= <TEXT> ; for use by human user
- Headers ::= <hName>“:”<hValue>
  - hName – Specific Header Name h (Attribut-Name)
  - hValue – Value of the value space of header h (Attribut-Wert)
- Data ::= <TEXT>

HTTP Request



HTTP Response



Typical HTTP Methods (1)

- Methods are applied in the context of HTTP Requests
  - For further detail on methods see RFC 2616
- **GET**
  - Deliver the resource addressed by the URI
- **POST**
  - Request to the server with respect to processing of encoded message body data (processing wrt the URI provided in POST)
  - Enables.:
    - Annotation of existing resources
    - Send (post) data blocks to an application, like comment, user ID, pictures
    - Available (send data via messages) from a form
- **HEAD**
  - Like GET but without the Response Body
- **OPTIONS**
  - Request on information submission on communication options

Typical HTTP Methods (2)

- **PUT**
  - Resources encoded in the Body should be saved at the Request URI
- **DELETE**
  - Server should remove the resources connected to the Request URI
- **TRACE**
  - Methods for development support of the so-called application layer request loop-back
  - All requests of user agents that the server gets should return to the user agent

Typical HTTP Headers (1)

- Repetition: Header entry (name-value pair) *hName:hValue*
  - The following examples are often used hNames
- **Content-Type**
  - Media type used
- **Expires**
  - Date/time from which the response is considered invalid
  - Important for caching!
- **Host**
  - Specifies Internet host and port number of the requested resourse
  - Required for "multi-homed server" (HTTP/1.1)!
- **Last-Modified**
  - Date and time when the "variant" (object referenced by the Request-URI) was last modified
  - Important for caching!

Typical HTTP Headers (2)

- **Location**
  - Is set in the HTTP Response to notify the user agent of the new location of the requested Request-URI
  - Very important concept in different protocols which build up on HTTP, for example, in the security area
  - Is used for implementation of the so-called "Redirects"
- **Referrer**
  - Reference to the URI from which the user agent has posted the current Request URI
  - Useful for maintenance/service tasks
    - Where do the users come from?
    - Logging, optimization, caching, user types
  - Not used as a security mechanism
- **User-Agent**
  - Information about the user agent
  - Important for personalization and internationalization
- Numerous further attributes exist for use for different tasks

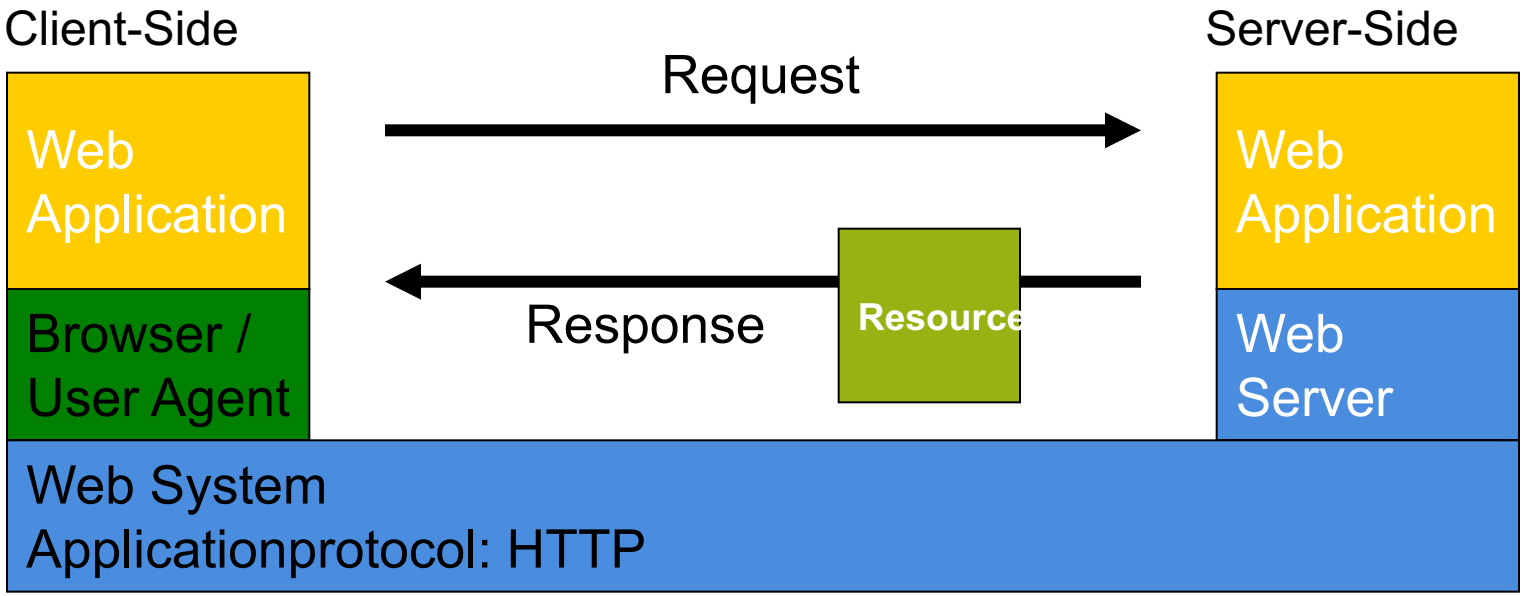
Typical HTTP Headers (3)

Code	Description
1xx	**Information as intermediate response**
2xx	**Successful operations**
200	OK
201	Created
3xx	**Redirects**
301	Moved Permanently
302	Moved Temporarily
4xx	**Client Error**
400	Bad Request – not understood
401	Unauthorized
403	Forbidden – not authorized
404	Not Found
5xx	**Server Error**

Section

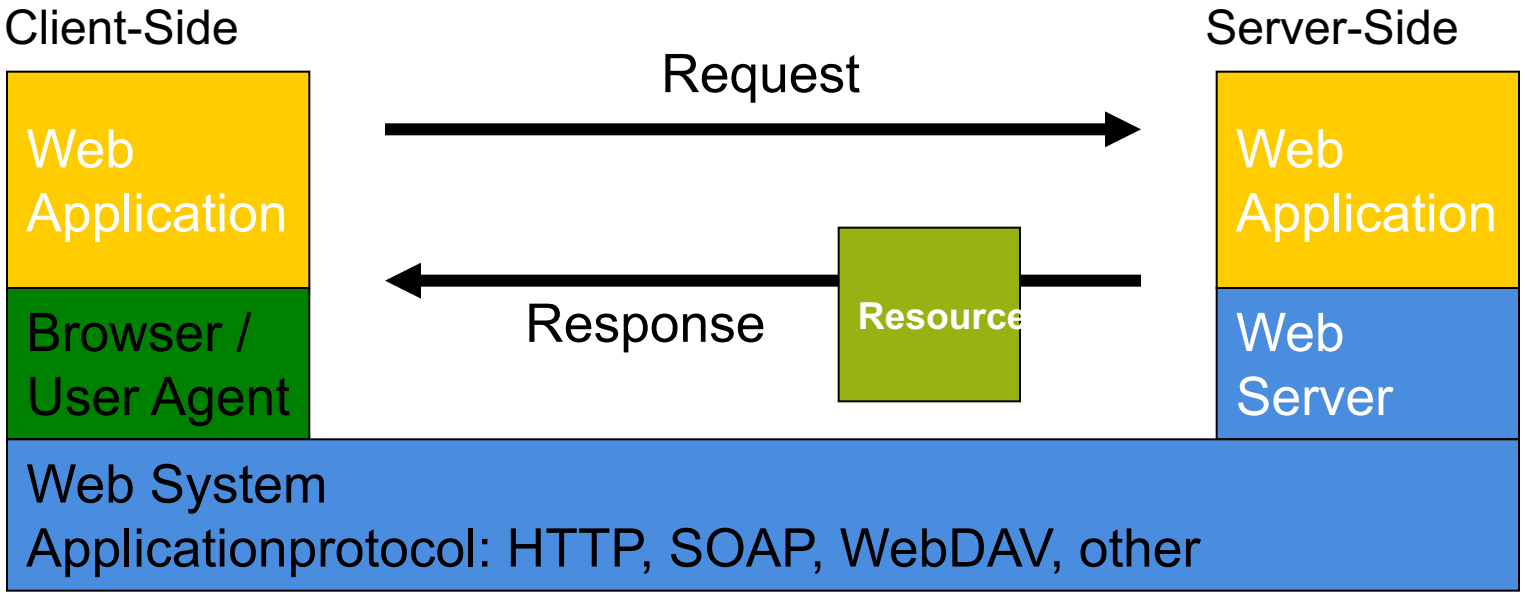
WORLD-WIDE-WEB EVOLUTION

2nd Generation World-Wide Web



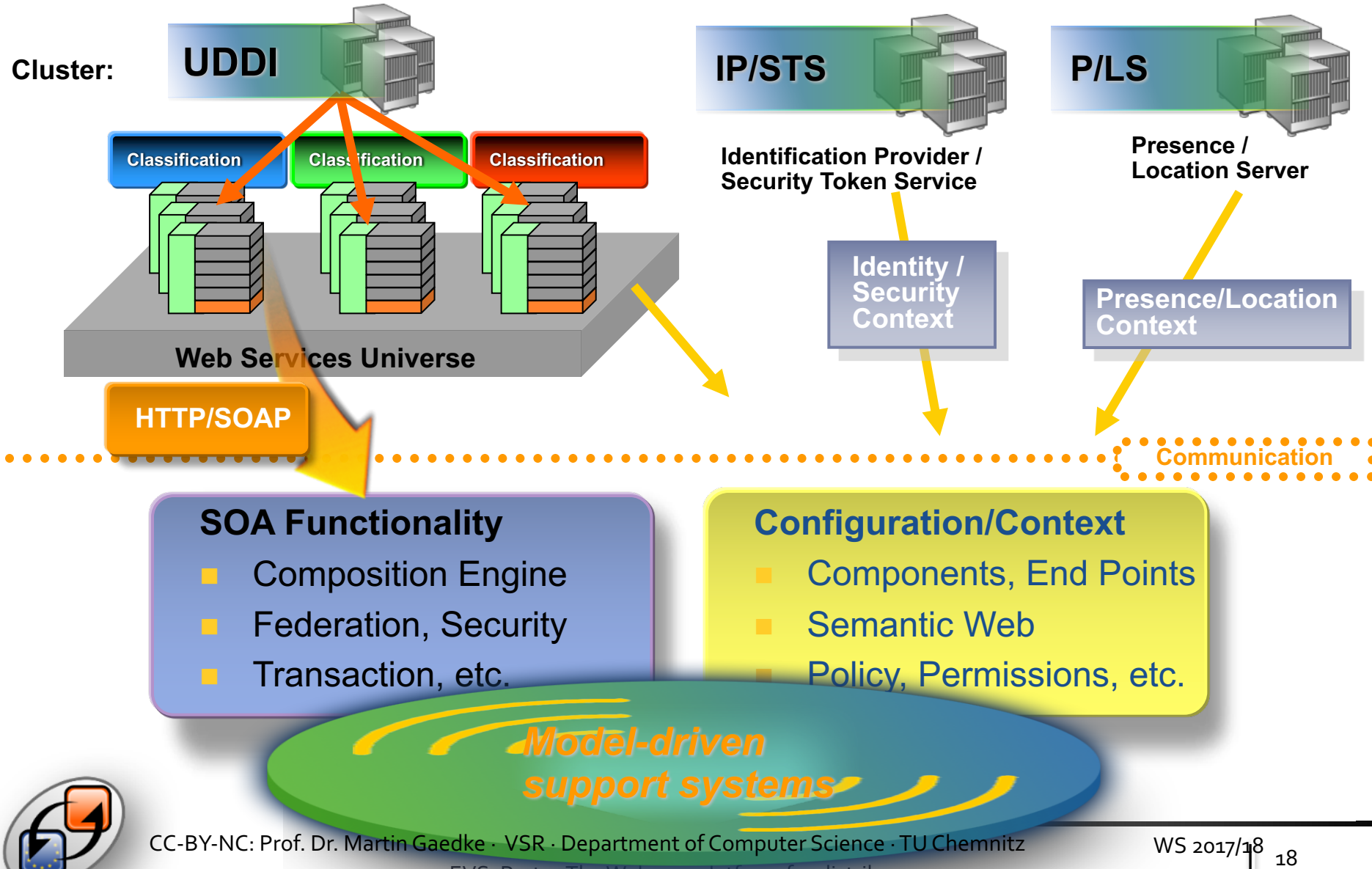
- Browser
  - Mosaic, Netscape
  - HTML, Frames
  - Images
  - HTML-Forms
  - Helper
    - Audio, Video etc.
- Web System
  - HTTP
  - Cookies
- Web Server
  - HTTP
  - Server-API & CGI
    - Database
    - Information Systems
    - Media Server

3rd Multi-Tier Gen World Wide Web



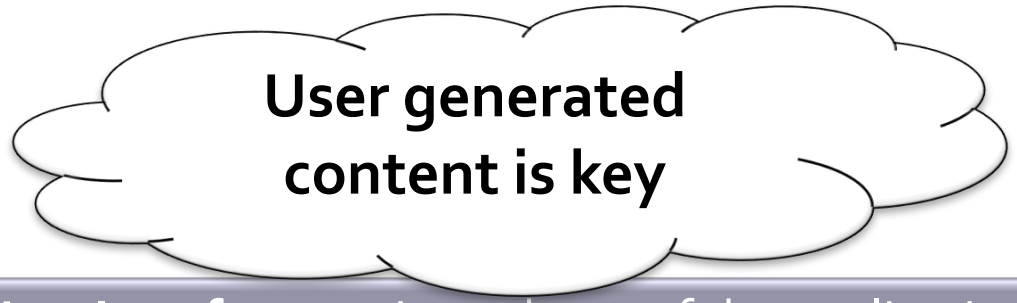
- User Agent
  - Netscape, IE, and PDA-Browser etc.
  - Other Types of User Agent
  - Plug-Ins, Applets, ActiveX
  - Script-Code
  - DHTML, More…
- Web System
  - HTTP, WebDAV, SOAP, other
  - Cookies
  - UDDI
  - Other relevant protocols FTP, SMTP
  - More…
- Web Server
  - HTTP, more
  - Server-API & CGI
  - XML-Support
  - Component-Support
    - Servlets
    - Web-Services

4th Generation (SOA-buzz starting 2000)



5th Generation (around 2004)

Distributed application (take crowd into account)



User Interface – oriented part of the application
UI/UX & Interaction & Navigation & Client-side code & Sensor-code

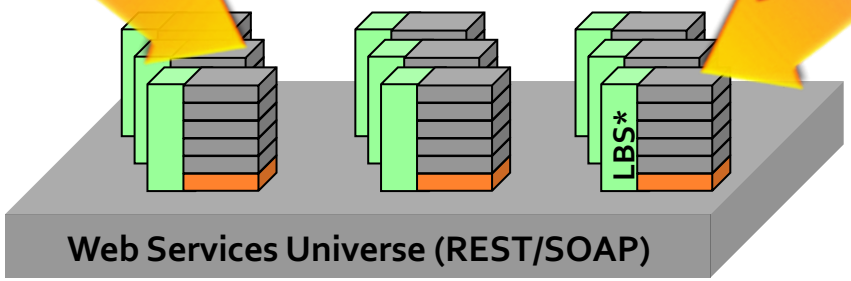
Browser (several)

Embedded User Agents

Mobile Phones

Communication

API-First Principle



several Identity Systems

*Location-based Service

6th Generation (around 2005)

Distributed application (take crowd into account)

User relationships are key

Social Web – oriented part of the application
(take social graph into account)

User Interface – oriented part of the application
UI/UX & Interaction & Navigation & Client-side code & Sensor-code

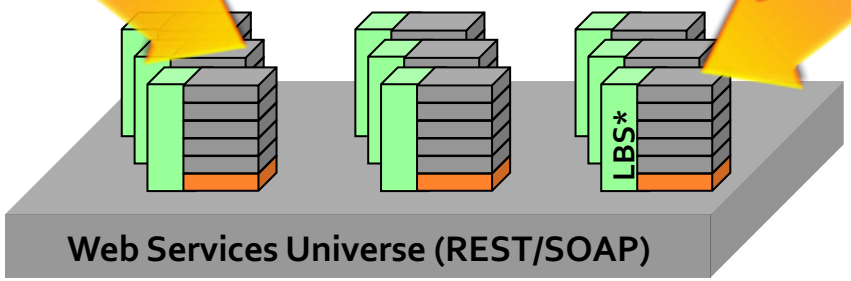
Browser (several)

Embedded User Agents

Mobile Phones and other devices (Tablets)

Communication

API-First Principle



several Identity Systems

*Location-based Service

7th Generation (around 2007)

Distributed application (take crowd into account)

User relationships are key

Social Web – oriented part of the application
(take social graph into account)

User Interface – oriented part of the application
UI/UX & Interaction & Navigation & Client-side code & Sensor-code

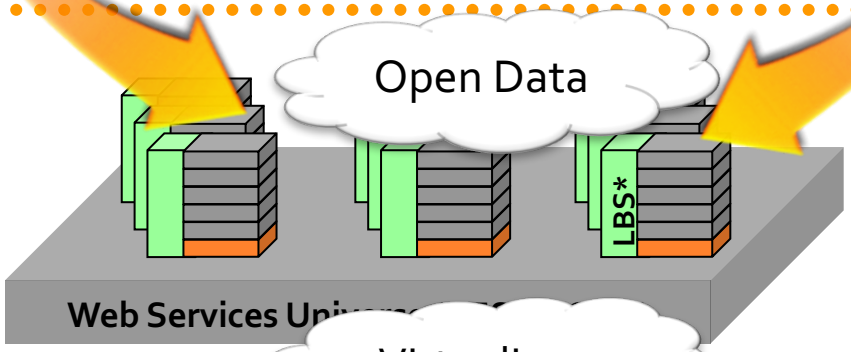
Browser (several)

Embedded User Agents

Mobile Phones and other devices (Tablets)

Communication

API-First Principle



several Identity Systems

*Location-based Service

8th Generation (around 2011)

Emotional relationships are key

Distributed application (take crowd into account)

Gamification oriented part of the application

Social Web – oriented part of the application

User Interface – oriented part of the application
UI/UX & Interaction & Navigation & Client-side code & Sensor-code

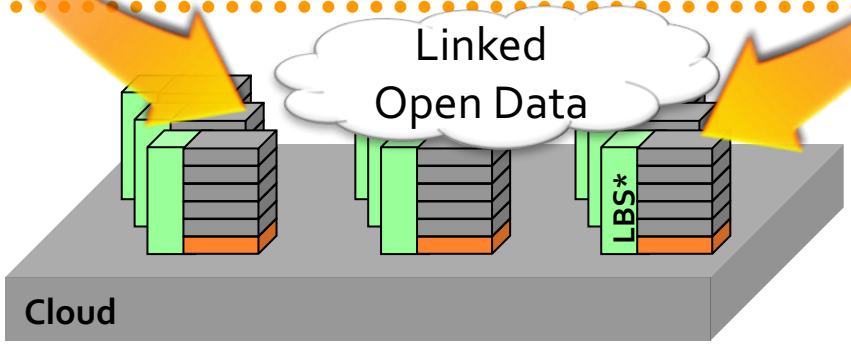
Browser (several)

Embedded User Agents

Mobile Phones and other devices (Tablets)

Communication

API-First Principle



Cloud-based Identity Systems

*Location-based Service

After 2011? Even faster evolution