Naming and Web Services

What is a problem, when designing a RPC-style Web Service for a Use Case like this? What are the disadvantages of SOAP?

CRUD heavy Applications are a bad use case for RPC/SOAP

* We have to define a lot of (CRUD) operations
* Repeatedly, we have to define the same (CRUD) operations for different object types
* We have to define a lot of messages, that contain no parameters or only one primitive
* SOAP is depending on XML; large overhead for complex objects

Comparing the REST API to the wsdl API of Exercise 3. What do you notice? Why is REST the better choice, especially for this Use Case?

* REST is data centric, therefore a nice Use Case for CRUD heavy applications
* Resources are identified by URL, unified schema
* Access always by the same operations POST, GET, PUT, DELETE
* Results in clear APIs; REST is independent of data representation

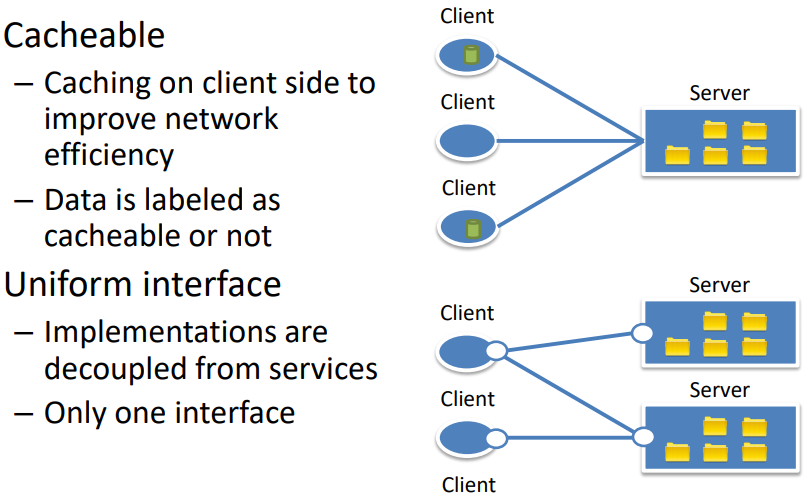
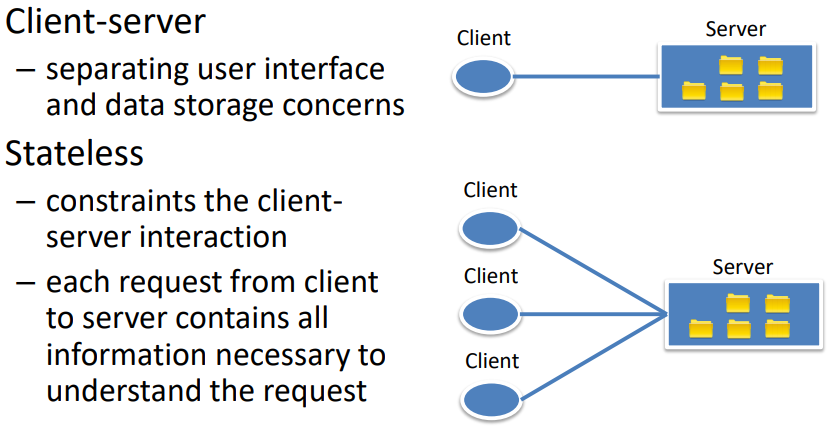
Why do root and zone servers always operate in iterative mode?

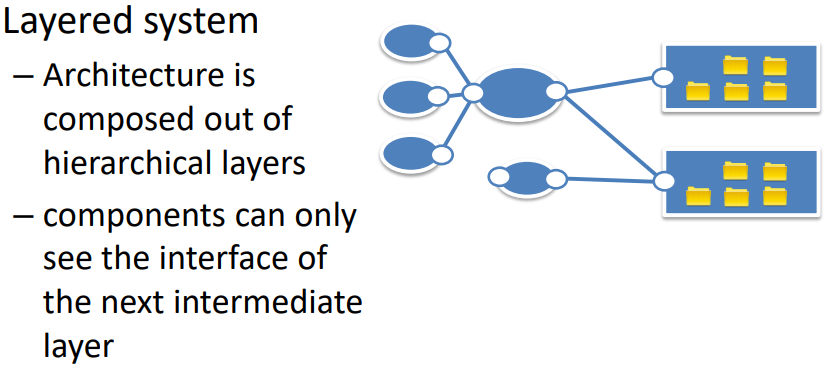
* Root and zone servers are the entry point of the DNS hierarchy. Thus, they are the most frequently requested servers.

Name two possible attacks on DNS?

* DNS amplification attack
* DNS cache poisoning

Rest styles



Rest operations

Interface provides four HTTP methods to manipulate resources

* **GET** (@GET, @Path(“/shoppingcart/items/{item.id})🡪 /shoppingcart/Ball/15
  + Request a resource from the server
* **POST** 🡪Creates new (sub) – resource below the specified resource
* **PUT** 🡪Specified resources will be created
* **DELETE** 🡪Deletes specified resource

REST advantages

* Easy to implement, simple design
* Scalable, uses HTTP caching
* Global identification of resources
* REST is the architectural style of the web itself
* REST is protocol independent

ES FEHLT SOAP