

Web-Based Systems

INF 5040 autumn 2015

lecturer: Roman Vitenberg

INF5040, Roman Vitenberg

1

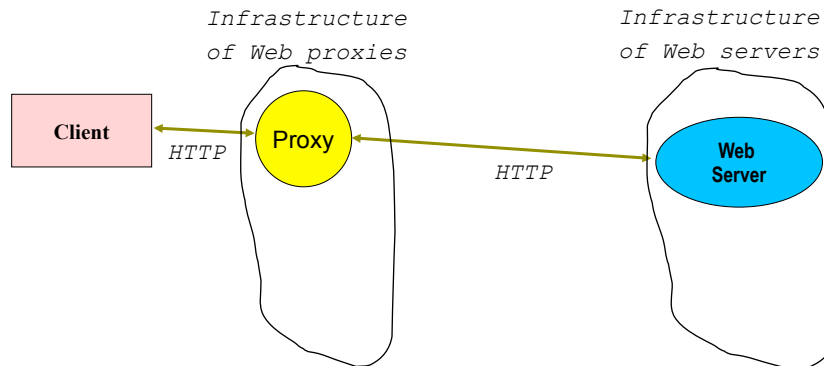
Two main flavors

- Browser-server WWW application
 - Geared towards human interaction
 - Not suitable for automation
 - Automatic restocking from amazon.com
 - Sniping in eBay
- Web services middleware
 - Generic extension of the WWW application
 - Web servers announce and provide services
 - Web server can be a client of another service

INF5040, Roman Vitenberg

2

Communicating entities



INF5040, Roman Vitenberg

3

Hypertext Transfer Protocol (HTTP)

- A simple document transfer protocol
 - A client sends a request & waits for a reply
 - GET,PUT,DELETE,HEAD, and POST methods
- Stateless
- Uses TCP as the underlying protocol
- In HTTP 1.0, each request was sent on a separate TCP connection
- HTTP 1.1 introduced **persistent connections** and **pipelining**
- A response may redirect the client to another document

INF5040, Roman Vitenberg

4

Web proxy functions

- Protocol translation and conversion
 - Not needed for modern browsers/clients
- Filtering requests and responses
- Logging
- Compression
- Caching

INF5040, Roman Vitenberg

5

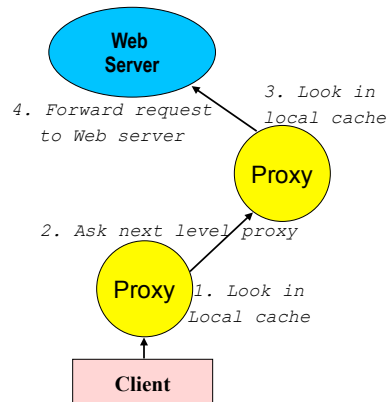
Client-side and proxy caches

- Cache update protocols
 - Pull with if-modified-since GET HTTP header
 - Lease-based propagation
- Not as effective for dynamic content
- Cache replacement policies
 - LRU is commonly used and it performs well
 - A number of specialised policies, e.g., Greedy-Dual

INF5040, Roman Vitenberg

6

Hierarchical proxy caching

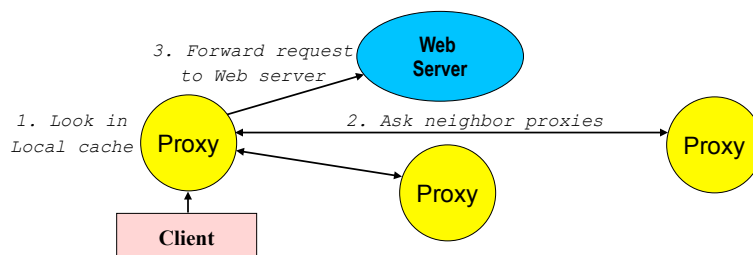


- Commonly deployed throughout the world
- Reduces latency and traffic for popular documents
- Increases latency for other documents
- Higher-level proxies require a lot of storage space

INF5040, Roman Vitenberg

7

Cooperative proxy caching

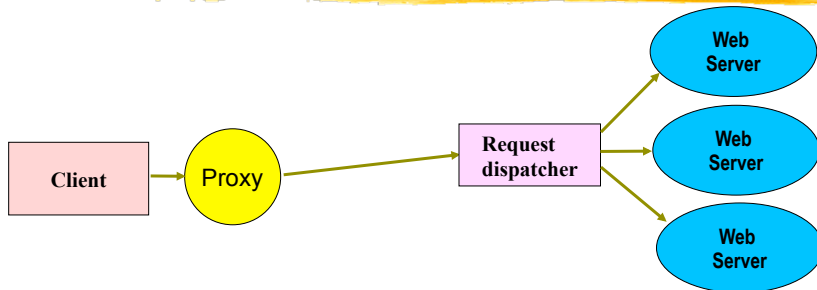


- Deployed on a per-organisation basis
- Can be combined with hierarchical caches

INF5040, Roman Vitenberg

8

Simple server-side backend



- Several possibilities for request dispatching
 - Round-robin DNS
 - Content-aware dispatcher inspecting HTTP requests
 - TCP-level switch

INF5040, Roman Vitenberg

9

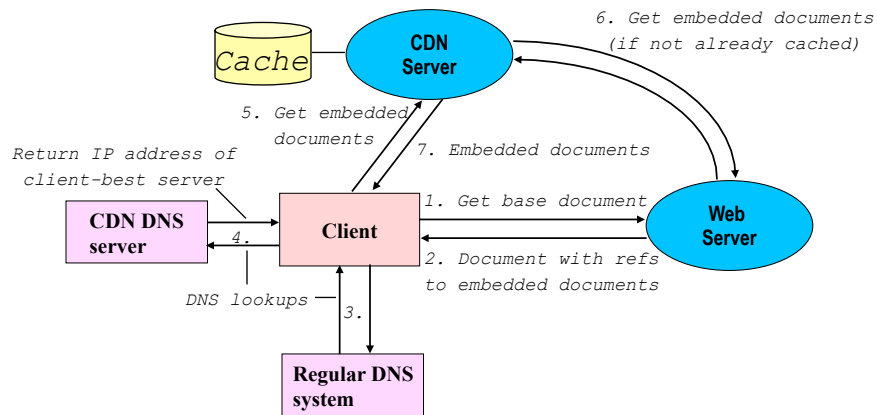
Content-distribution network (CDN)

- Placement of data/object replicas
 - See the lecture on replication...
 - A number of evaluation metrics
 - Latency (real-time and the number of hops)
 - Bandwidth (available and network usage)
 - Financial
- Consistency enforcement
 - See the lecture on replication...
- Redirection of client requests

INF5040, Roman Vitenberg

10

Redirection of client requests in an Akamai CDN



INF5040, Roman Vitenberg

11

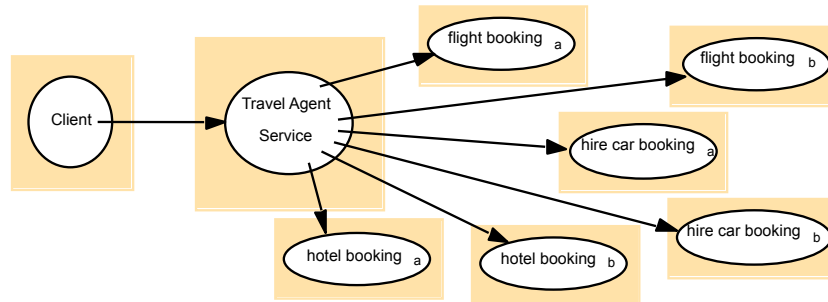
Web services

- Allows an application-specific client to communicate with a service over a functionality-specific API
- Allows a client program in an organisation to communicate with a server program in another
- Allows complex applications that integrate and compose services from other services
- Because the interaction is generic Web-services cannot be directly accessed by browsers

INF5040, Roman Vitenberg

12

Example from the Coulouris book



INF5040, Roman Vitenberg

13

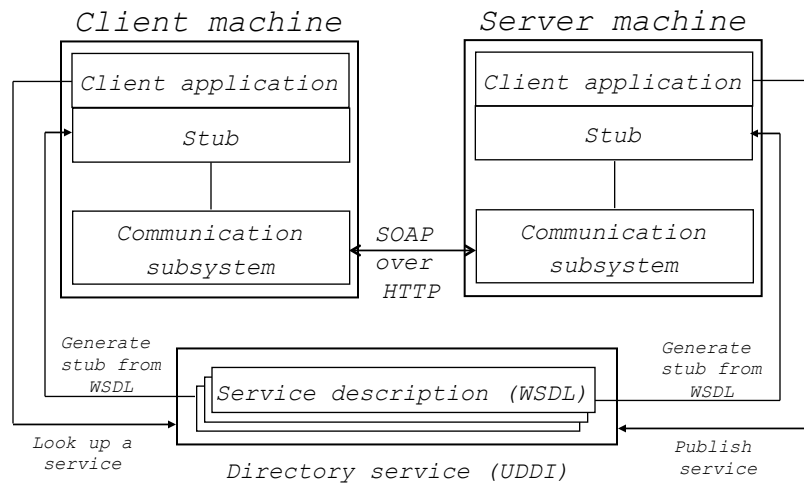
Applications of Web Services

- Many well-known web sites offer a Web-services interface to their clients
 - Amazon, Yahoo, Google, eBay...

INF5040, Roman Vitenberg

14

Architecture of Web Services

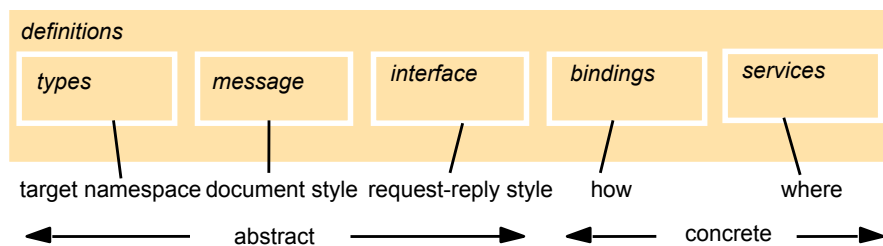


INF5040, Roman Vitenberg

15

WSDL description

Description of a service



INF5040, Roman Vitenberg

17

WSDL

- Types
 - Name and type of exchanged values (similar to declaration of variables)
- Messages (operations)
- Interface
 - Operation parameters
 - In-Out, In-Only, Out-In, Out-only...
- Binding
 - The choice of communication protocol
 - typically SOAP, HTTP, or MIME, but others are also possible (e.g., GIOP in order to communicate with Corba)
- Services
 - Specific endpoint addresses, one for each binding
 - For the SOAP binding, it will be a URI of the service location

INF5040, Roman Vitenberg

18

URI – Uniform Resource Identifier

- URL is the simplest form of URI
 - Based on DNS names
 - Partly location-dependent (in any case dependent on domain names of the machine)
- URN (Uniform Resource Names) is another possibility that is location independent
 - But they are also more prone to name clashes
 - Dependent on Directory Service – “Universal Directory and Discovery Service UDDI”

INF5040, Roman Vitenberg

19

XML – Extensible Markup Language

- A language for describing message formats
- Defines how the message is parsed
- UNICODE-based
 - Readable by both human and machines
 - Ineffective space-wise
- A language that is suitable to represent a hierarchical data structure in a flat UNICODE-based form

XML definition example

```
<person id="123456789">  
  <name>Smith</name>  
  <place>London</place>  
  <year>1934</year>  
  <!-- a comment -->  
</person >
```

Namespace in the *Person* structure

```
<person pers:id="123456789"
      xmlns:pers = "http://www.cdk4.net/person">
  <pers:name> Smith </pers:name>
  <pers:place> London </pers:place >
  <pers:year> 1934 </pers:year>
</person>
```

INF5040, Roman Vitenberg

22

Schema for the *Person*-structure

```
<xsd:schema xmlns:xsd = URL of XML schema definitions >
  <xsd:element name= "person" type = "personType" />
  <xsd:complexType name= "personType">
    <xsd:sequence>
      <xsd:element name = "name" type="xs:string"/>
      <xsd:element name = "place" type="xs:string"/>
      <xsd:element name = "year" type="xs:positiveInteger"/>
    </xsd:sequence>
    <xsd:attribute name= "id" type = "xs:positiveInteger"/>
  </xsd:complexType>
</xsd:schema>
```

INF5040, Roman Vitenberg

23

SOAP – Simple Object Access Protocol

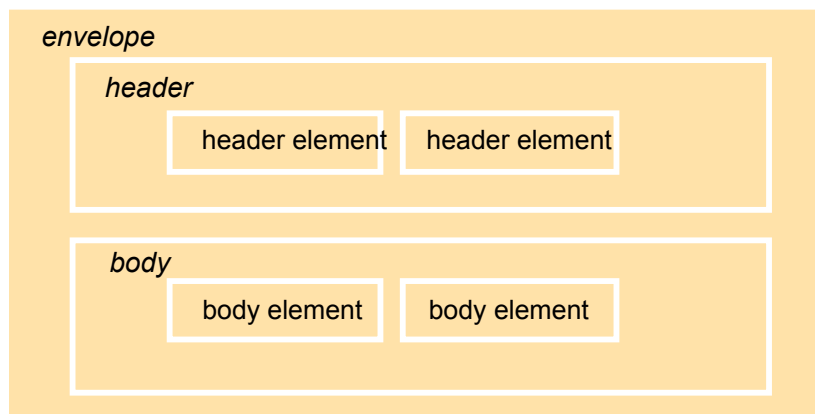
- Defines how XML should be used to represent the content of messages
- Defines how a pair of messages can form a request/reply template
- Rules regarding how message recipient should process contained XML-elements
- How HTTP should be used for exchanging SOAP messages

INF5040, Roman Vitenberg

24

SOAP Message

Envelope, Header, and Body



INF5040, Roman Vitenberg

25

SOAP message without the header

env:envelope xmlns:env = namespace URI for SOAP envelopes

env:body

m:exchange

xmlns:m = namespace URI of the service description

m:arg1

Hello

m:arg2

World

INF5040, Roman Vitenberg

26

SOAP-response

env:envelope xmlns:env = namespace URI for SOAP envelope

env:body

m:exchangeResponse

xmlns:m = namespace URI for the service description

m:res1

World

m:res2

Hello

INF5040, Roman Vitenberg

27

Example of a SOAP message embedded in HTTP

```

POST /examples/stringer ← endpoint address
Host: www.cdk4.net
Content-Type: application/soap+xml
Action: http://www.cdk4.net/examples/stringer#exchange ← action

```

HTTP header

```

<env:envelope xmlns:env= namespace URI for SOAP envelope >
<env:header> </env:header>
<env:body> </env:body>
</env:Envelope>

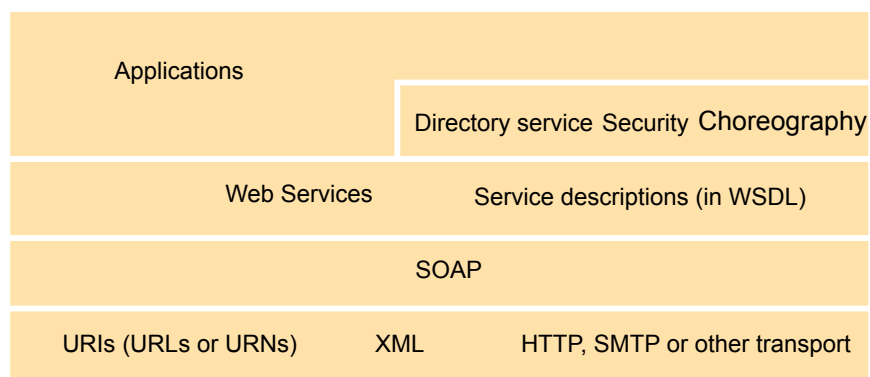
```

Soap message

INF5040, Roman Vitenberg

28

Infrastructure and components - summary



INF5040, Roman Vitenberg

29

Main differences from distributed objects

- Many similarities, but
- Some object concepts do not exist
 - Cannot instantiate and remove objects
 - Garbage collection is irrelevant
 - Using simpler Universal Resource Identifiers instead of Remote Object References
- “Web Services are not Distributed Objects” by Werner Vogels
- “Like it or not, Web Services are Distributed Objects” by Ken Birman

INF5040, Roman Vitenberg

30

Web services vs CORBA

- The scope of deployment
 - CORBA: inside one organisation or a consortium of mutually known collaborating organisations
 - Web Services: truly global
 - Difference in the naming and references
 - WS discovery is based on DNS that is scalable and global
- Deployment vs communication middleware
 - Corba: both
 - Web services: communication only
- Interoperability

INF5040, Roman Vitenberg

31

Web services vs CORBA (cont'd)

- Ease of use and the learning curve
 - Web Services
 - Based on HTTP and XML infrastructure that already exists in most operating systems and platforms
 - Messages are human-readable
 - Only additionally requires API for SOAP
 - CORBA: installation/administration + high learning curve
- Efficiency
 - Web Services:
 - Long messages that it takes time to parse
 - Message processing hundreds of times slower compared to CORBA