INFOH-511 WEB SERVICES

LECTURE 1: Introduction

COURSE RESPONSIBLES

- **Stijn Vansummeren**

  ULB, Campus Solbosch, UB4.125

  stijn.vansummeren@ulb.ac.be

- **Francois Picalausa**

  ULB, Campus Solbosch, UB4.133

  fpicalau@ulb.ac.be

PREREQUISITES

- This is an advanced Master level course. Students are expected to have the following required prior knowledge:
  - INFOH-509 XML & Web Technologies
  - Basic knowledge about computer networks
  - Basic programming skills (Java)

- It is possible to follow the course without these prerequisites, but beware that this implies an additional workload to obtain the required prior knowledge!

Course objectives

- To give an introduction to web services and their related technologies, in particular:
  - Obtain a working knowledge of the internet and the HTTP protocol
  - Obtain an understanding of the 2 major classes of web service technologies :
    - the "Big Web Services" (WS-*)
    - the "REST"-style web services ;
  - To critically analyze the advantages and disadvantages of both sets of technologies ;
  - To obtain a background on Service Oriented Architecture (SOA) and Resource Oriented Architecture (ROA) ;
  - To obtain a background on Business Processes and the Business Process Execution Language (BPEL).

Competences to develop

- After successful completion of this course you should be able to
  - build programming-language specific web service client wrappers given a web service to connect to, based on the working knowledge of HTTP, SOAP, WSDL, UDDI
  - analyze web service requirements, and choose the according appropriate technology for implementation
  - design and implement web services from the server side ;
  - critically evaluate new web service technologies.

COURSE ORGANISATION

- The course is organized as a mixture of:
  - Ex-cathedra lectures
  - Seminars prepared by, presented by, discussed with students
  - Technical exercises
  - Project work

- Seminar assignments, reading assignments, project assignment are all published on the course website:

  http://cs.ulb.ac.be/public/teaching/infoh511

- Check regularly for updates!

Course schedule

Date	Time		Content	Room
Fri 17 Feb	14h-16h	Lect. 1	HTTP + Web Service general intro	S.AW1.126
Mon 20 Feb	14h-17h	Ex. 1	HTTP request & programmatic HTTP request (REST)	UB4.126
Fri 24 Feb	14h-16h	Lec. 2	REST	S.H2111
Mon 27 Feb	14h-17h	Ex. 2	Lab 2 : Developping REST services	UB4.126
Fri 9 Mar	14h-16h	Lect/sem 3	SOAP	S.H2111
Mon 12 Mar	14h-17h	Ex. 3	Lab 3 : Using & developing SOAP services	UB4.126
Fri 16 Mar	14h-16h	Lect/sem 4	WSDL	S.H2111
		Project	*Project assignment*	
Mon 26 Mar	14h-17h	Ex. 4	Lab 4 : Composite services	UB4.130
Fri 30 Mar	14h-16h	Lect/sem 5	UDDI	S.H2111
Mon 16 Apr	14h-17h	Ex. 5	Lab 5 : Presentation layer	UB4.126
Thu 19 Apr	14h-16h	Lect/sem 6	Security 1	S.K.3.401
Mon 23 Apr	14h-17h	Ex. 6	Free to work on project. Opportunity for Q&A.	UB4.126
Thu 26 Apr	14h-16h	Lect/sem 7	Security 2	S.K.3.401
Mon 30 Apr	14h-17h	Ex. 7	Free to work on project. Opportunity for Q&A.	UB4.126
Thu 3 May	14h-16h	Lect/sem 8	Addressing, resources,notification	S.K.3.401
Mon 7 May	14h-17h	Ex. 8	Free to work on project. Opportunity for Q&A.	UB4.126
Thu 10 May	14h-16h	Lect. 9	Workflows & processes	S.K.3.401
Thu 24 May	14h-16h	Q&A	Q & A session	S.K.3.401
Fri 1 June		**Deadline**	**Project deadline**	

Examination

- The course does not have a traditional exam
- You are graded on:
  - Active participation to the seminars (4 / 20)
  - The preparation & presentation of one seminar (6/20)
  - Project work (10 / 20)

Seminars

- For each seminar, a small group of two to three students is given a list of reading resources on a particular topic as well as a set of accompanying questions.
- The group is asked to prepare a presentation on the topic (answering the given questions) and present this presentation during the seminar.
- The non-presenters are requested to read the same resources in preparation of the seminar ;
- During the seminar, the presenters present their presentation while the others critically evaluate this presentation, and fuel discussion on the answers presented to the questions.
- **Attending seminars is a mandatory requirement for passing the course!**

Seminar assignment example

Assignment Create and present a presentation on the topic of "SOAP". You should consult at least the following resources to prepare your presentation, but are encouraged to consult additional ones (e.g. on the Web, in the library, ...).

- M. P. Papazoglou, *Web Services: Principles and Technology*, chapter 4, pages 119–145.
- World Wide Web Consortium, *SOAP version 1.2 Part 0: Primer* http://www.w3.org/TR/soap12-part0/.
- A. Skonnard, *Understanding SOAP*, available on Microsoft MSDN http://msdn.microsoft.com/en-us/library/ms995800.aspx

Topics to discuss Your presentation should answer the following questions (possibly in a different order):

- What is SOAP?
- What problem does it intend to solve?
- How has SOAP evolved historically? What is the latest version of SOAP? What is the difference with prior version(s)?
- What is the structure and semantics of a SOAP message?
- How does SOAP deal with errors?
- What is the SOAP processing model? (SOAP node, SOAP message path, roles)

Group distribution

Seminar	Date	Students
SOAP	9 MARCH	
WSDL	16 MARCH	
UDDI	30 MARCH	
SECURITY 1	19 APRIL	
SECURITY 2	26 APRIL	
ADDRESSING, RESOURCES, NOTIFICATION	3 MAY	

TODO? Send by email / fix groups

WEB SERVICE = SERVICE ON THE WEB

- What is the Web?
- What is a Service?

THE WEB:
BASIC ARCHITECTURE & PROTOCOLS

The Web ≠ the Internet

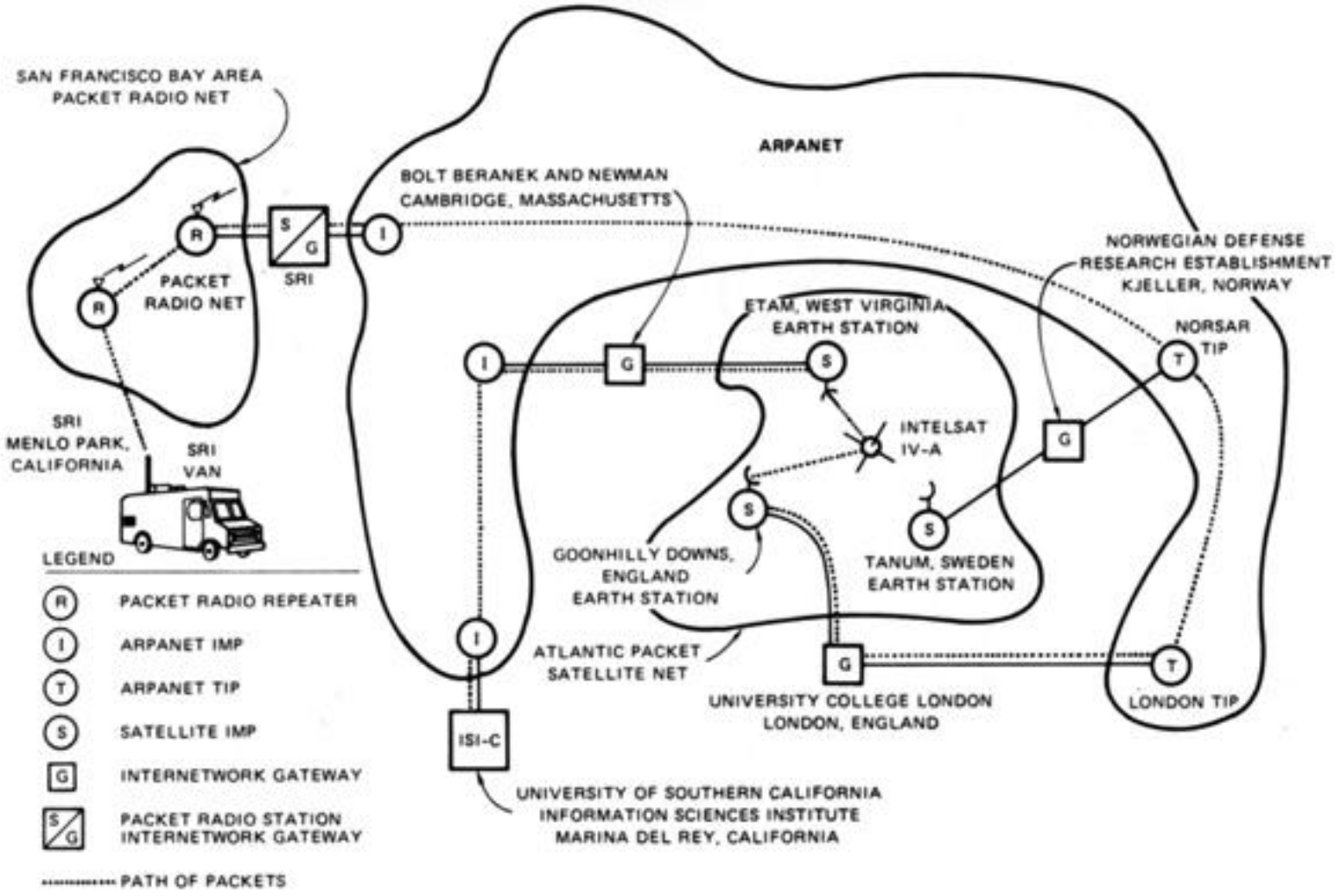
- The Internet is a global system of interconnected computer networks.

- The Web is a subset of the Internet. It is a collection of resources, linked by hyperlinks and URLs, transmitted by web browsers and web servers talking HTTP.

The Internet: some history

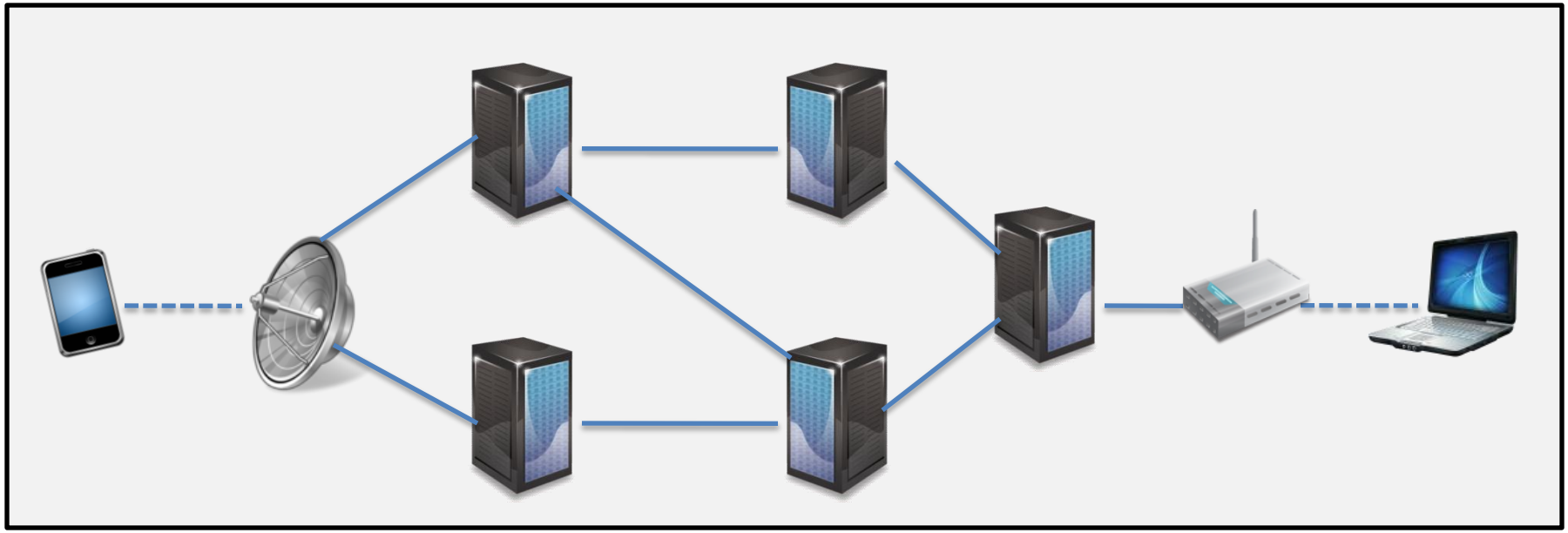
- **ARPANET (1969):**
  Advanced **R**esearch **P**roject **A**gency **NET**work
  - Created by the US Department of Defense
  - First operational **packet switching** network
  - Early ARPANET applications
    - Email, SMTP (1971), Ray Tomlinson
    - File Transfer Protocol, FTP (1973)
- Standardized **Internet Protocol Suite** (**TCP/IP**, 1983)

First ARPANET Logical Map



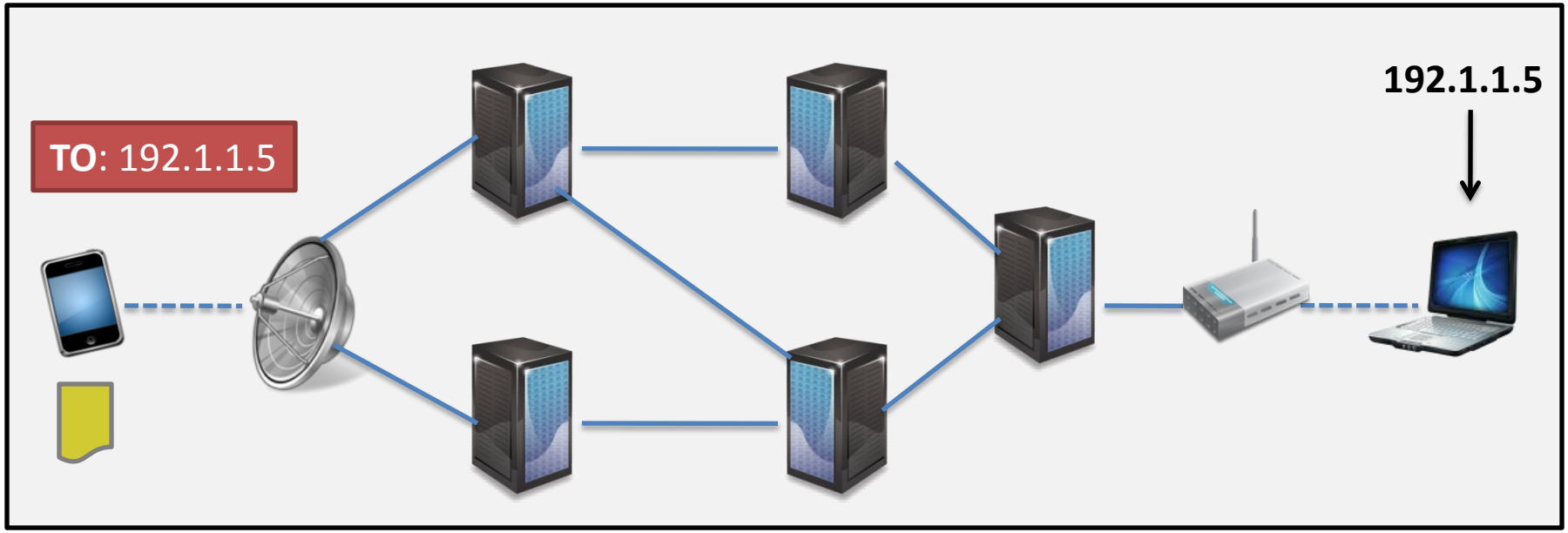
The Internet: simplified operation

- The Internet is a global system of interconnected computer networks.
- Computers that have a “physical link” (ethernet wire, WIFI, satelite, …) can talk directly to each other.
- Messages between non-adjacent computers are routed through adjacent computers to the destination.

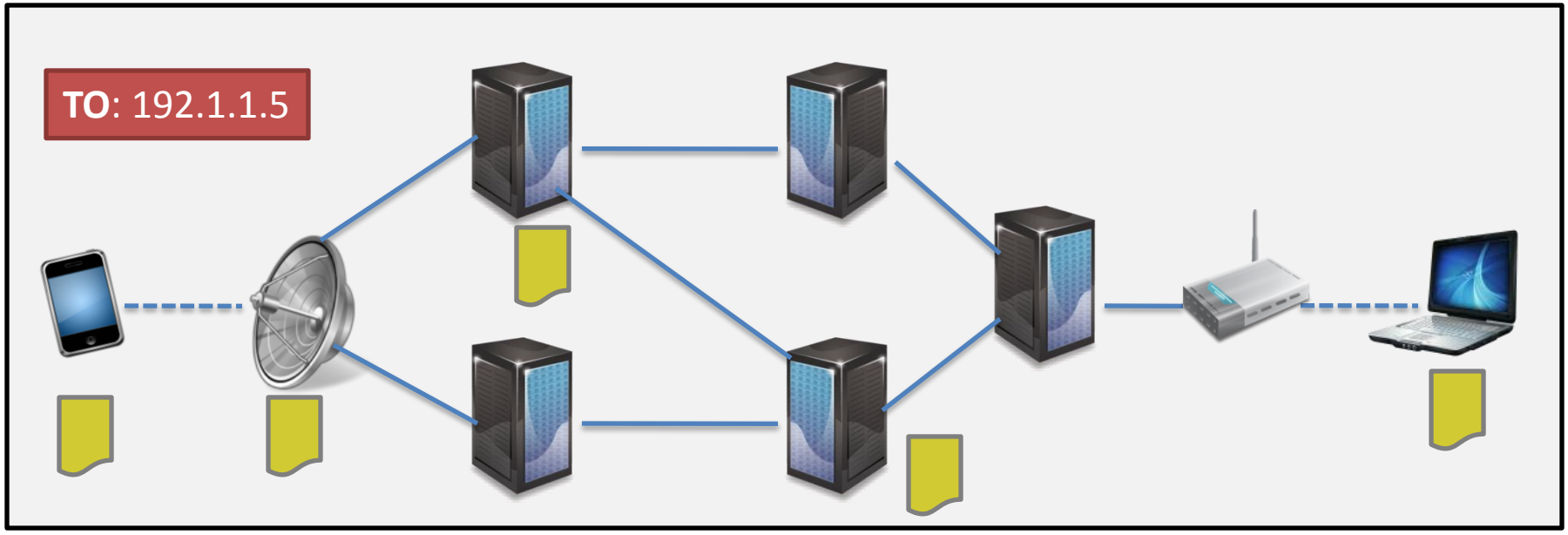


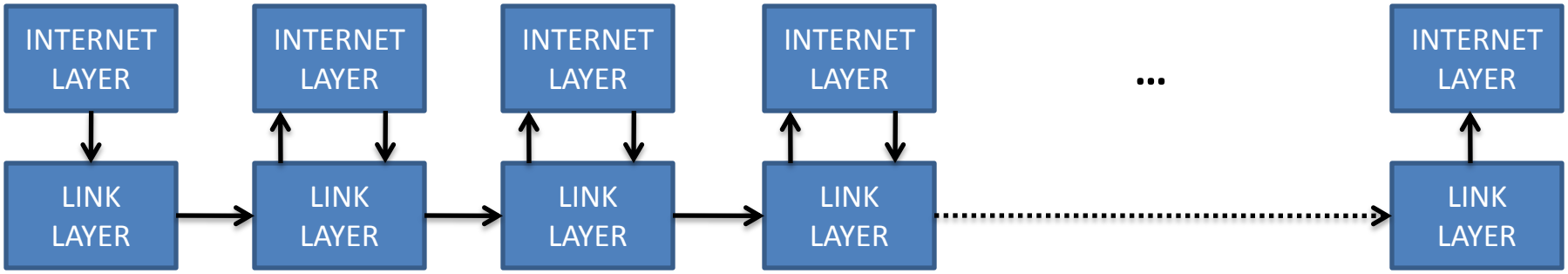
IP: Internet Protocol

- IP allows unreliable communication of *limited size data packets (called datagrams)* between machines identified by *IP addresses* (e.g., 164.15.59.215)
- Each datagram is sent and routed independently (and may arrive out of order). Routing is done transparently by the protocol.



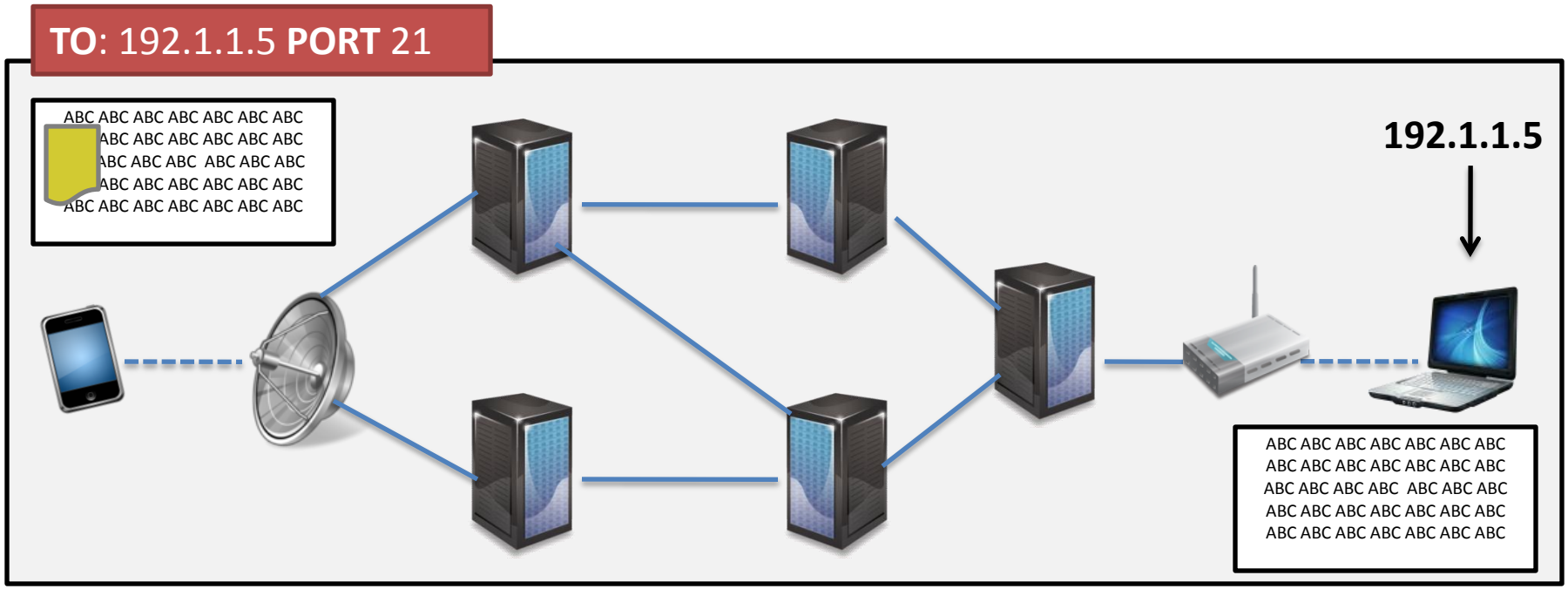
IP: Internet Protocol [Dataflow]





TCP: Transmission Control Protocol

- Transmission of arbitrary-length data in streams
- Transparently cuts up streams into fixed-size IP datagrams, and uses IP to send them to destination.
- Builds a reliable bi-directional communication channel on top of IP by retransmitting lost datagrams, reordering, etc.



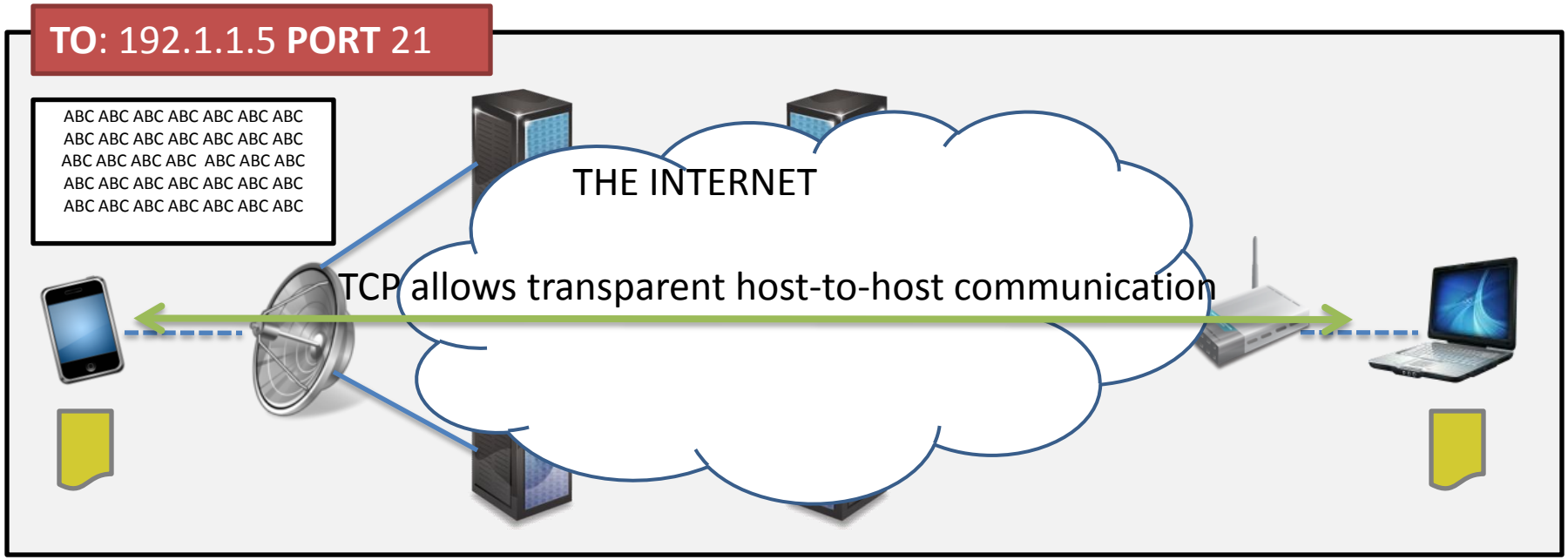
TCP: Transmission Control Protocol

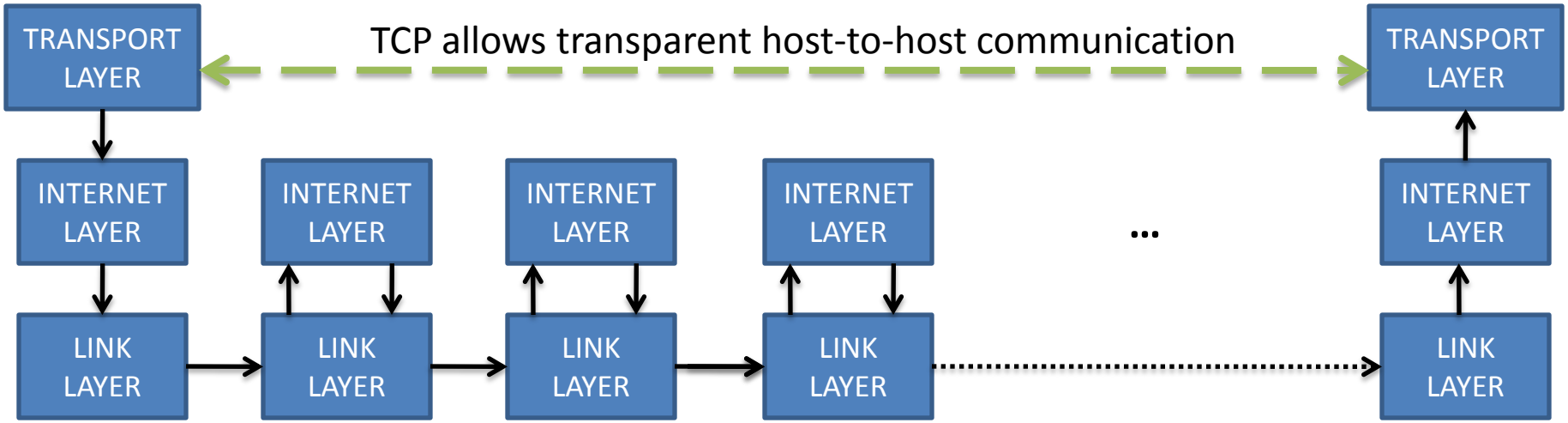
- **TCP is Connection-oriented**
  - Establish connection between client and server process
  - Transmit data in both directions
  - Close connection
- End point of connection given by a pair

IP address : port number

- A port numbers identifies the server/client *process* for which the data is intended
- Standard services (email, web browsing, ftp) have a fixed port number (see http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml)

TCP [Dataflow]

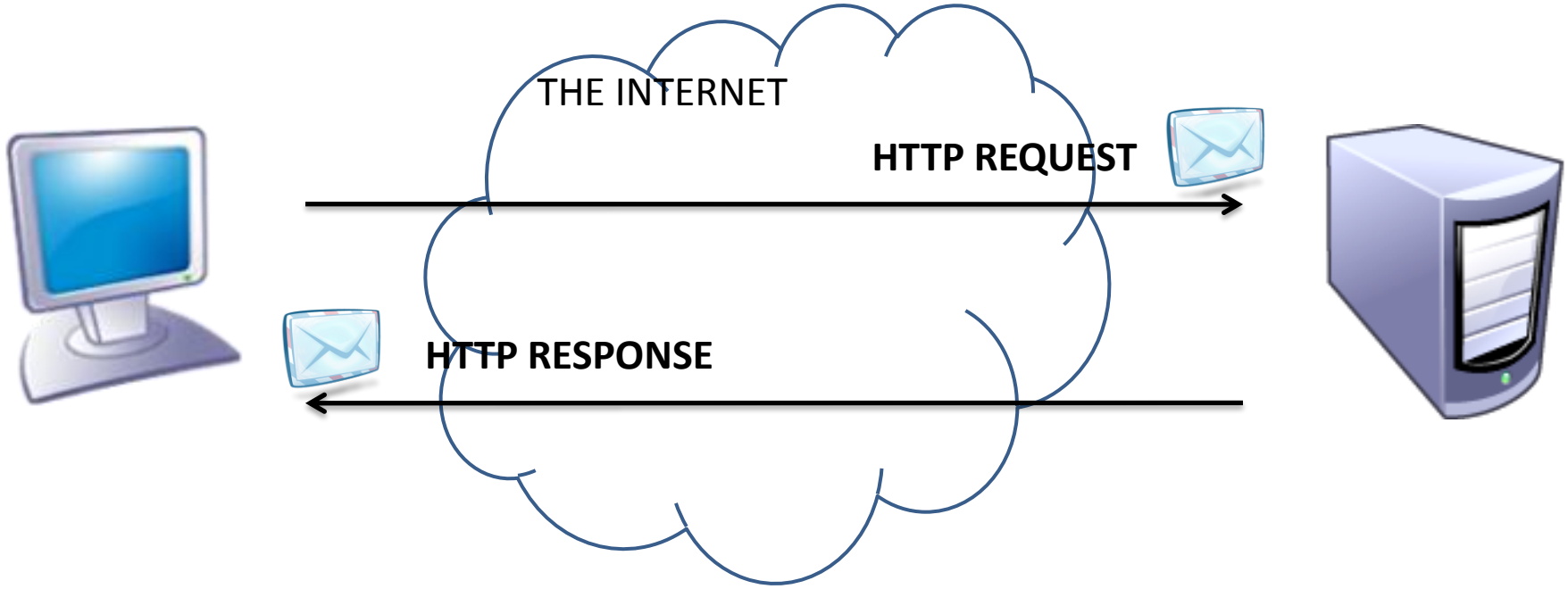




HTTP: Hypertext Transfer Protocol

- Layer on top of TCP
- Essentially an *envelope format*
- Request and response communication protocol for exchanging web resources (e.g., HTML, XML, TEXT, …)
- Communication is always initiated by the client
- Server typically runs on port 80
- Stateless, light-weight

HTTP = Request/Response protocol



Effect of typing http://www.ulb.ac.be/index.html in web browser :

1. Use a **domain name service (DNS)** to get the IP address for www.ulb.ac.be
2. Create a TCP connection to address 164.15.59.215 on port 80
3. Send a HTTP request message over the TCP connection
4. Receive the HTTP response (and visualize in a browser)

Example HTTP Request message


```
GET /index.html HTTP/1.1
Host: www.ulb.ac.be
User-Agent:
Accept: text/html,application/xhtml+xml,application/xml
Accept-language: text/xml,application/xml,text/html;q=0.9,…
Accept-encoding: us,en;q=0.5
Accept-charset: ISO-8895-15,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
```

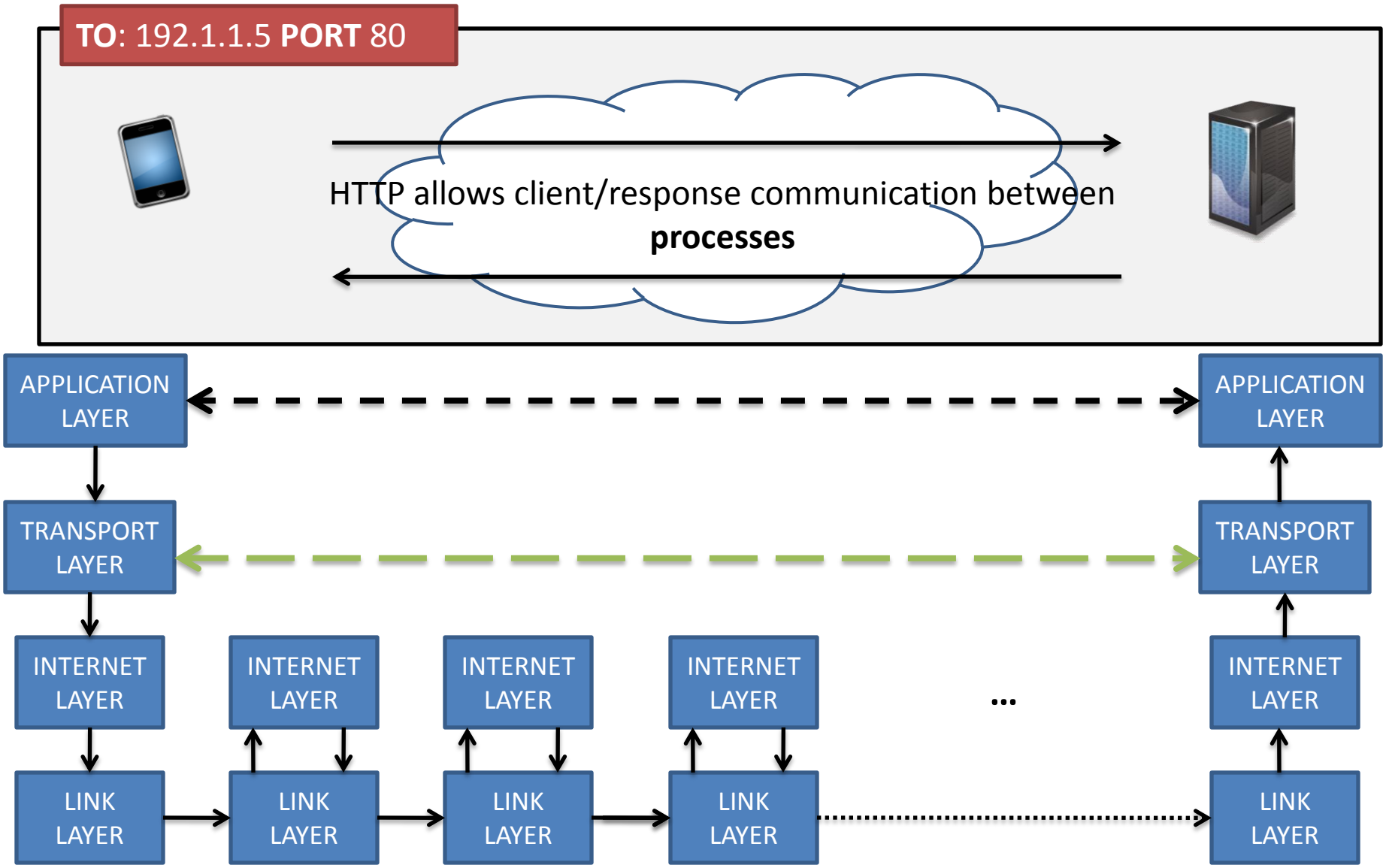
Example HTTP Response message


```
HTTP/1.1 200 OK
Date: Mon, 19 Dec 2011 16:39:16 GMT
Server: Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.7d
Last-Modified: Mon, 19 Dec 2011 15:48:25 GMT
ETag: "2725d3-11441-4b473e18e0130"
Accept-Ranges: bytes
Content-Length: 70721
Content-Type: text/html

<html xmlns=http://www.w3.org/1999/xhtml> … </html>
```

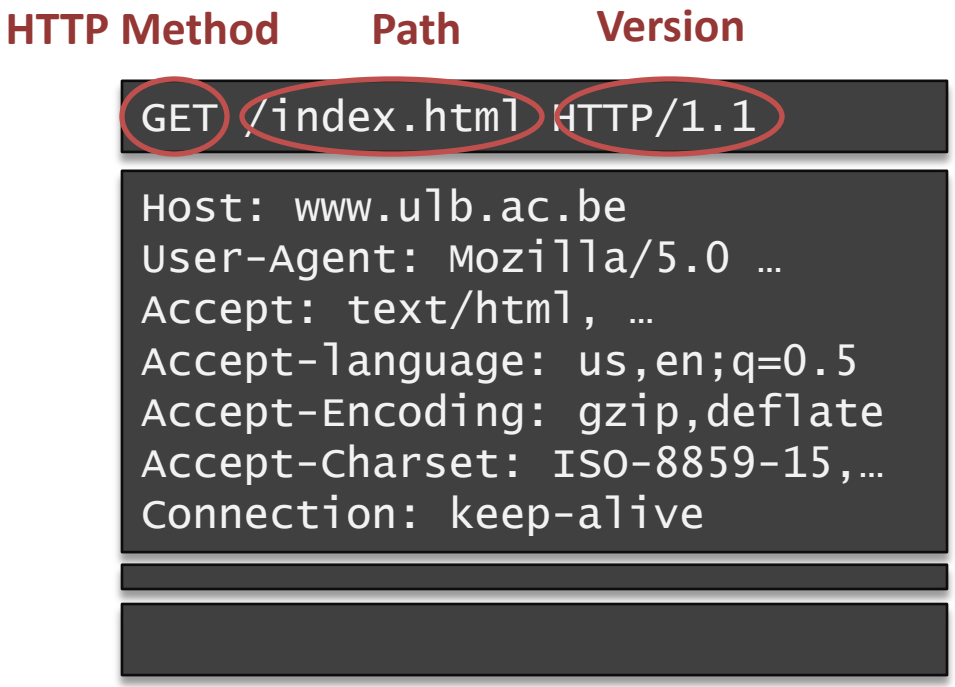

Demo

HTTP [Dataflow]



General structure of a HTTP request

- HTTP is a document-based protocol: the client puts a document in an envelope and sends it to the server
- The server replies with a response document in an envelope
- HTTP defines what the envelope should look like, but doesn't care what goes inside



Start Line

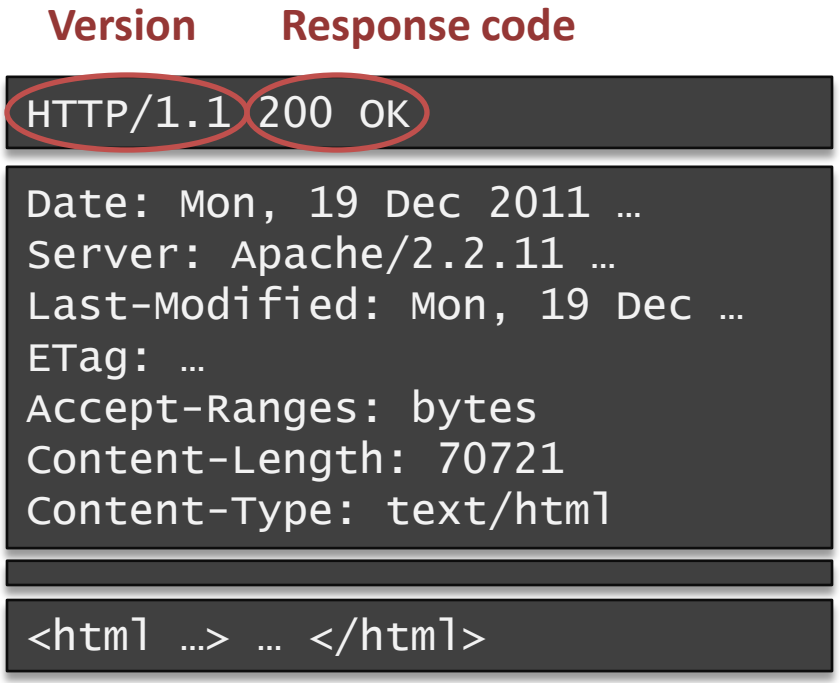
Request Headers
(= a list of key/value pairs)
"the stickers on the envelope"
Many standard headers
User-define headers possible

Empty line (CRLF)

The optional entity-body (document, resource representation)

General structure of a HTTP response

- HTTP is a document-based protocol: the client puts a document in an envelope and sends it to the server
- Server replies with a response document in an envelope
- HTTP defines what the envelope should look like, but doesn't care what goes inside



Status Line

Response Headers
(= a list of key/value pairs)
"the stickers on the envelope"
Many standard headers
User-define headers possible

Empty line (CRLF)

The optional entity-body (document, resource representation)

HTTP Methods [request only]

- The HTTP methods that can be used in a request are:

GET	request a resource representation
POST	send data to server and receives result
PUT	create or update a resource
DELETE	delete a resource
OPTIONS	Discover what HTTP methods are supported at target URI
HEAD	requests headers only (similar to GET but omits entity body)

- Most websites use only GET

The message body [request + response]

- The body is a sequence of bytes
- If a HTTP message includes a body, there is usually a header line that describes the MIME format of the body

```
Content-Length: 70721
Content-Type: text/html
```

Length in bytes

MIME type

(others include image/gif, application/xml, application/json, …)

- The MIME type identifies how the entity body should be interpreted.
- Full list of MIME types: http://www.iana.org/assignments/media-types/index.html

HTTP response codes [response only]

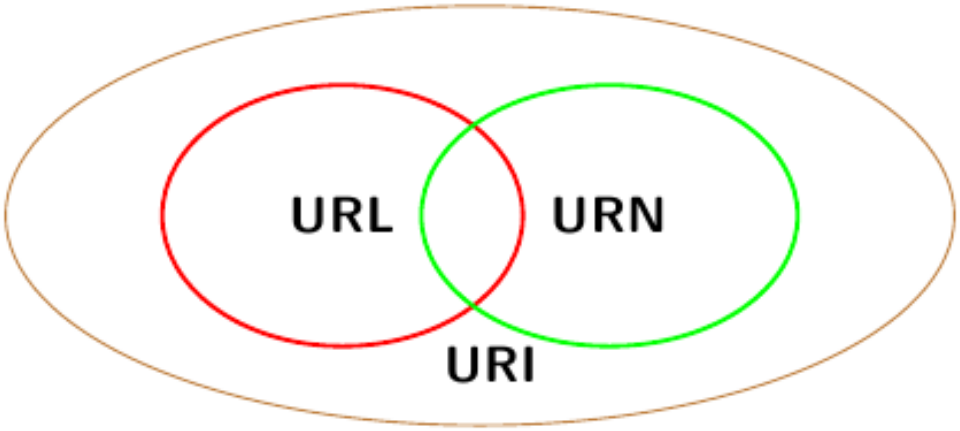
- The response code is a three-digit integer, where the first digit identifies the general category of response:
  - 1xx indicates an informational message only
  - 2xx indicates success of some kind
  - 3xx redirects the client to another URL
  - 4xx indicates an error on the client's part
  - 5xx indicates an error on the server's part
- The most common status codes are:

```
200 OK
```

```
404 Not Found
```

- The code is followed by a human-readable phrase, which may vary from server to server

A note on URIs

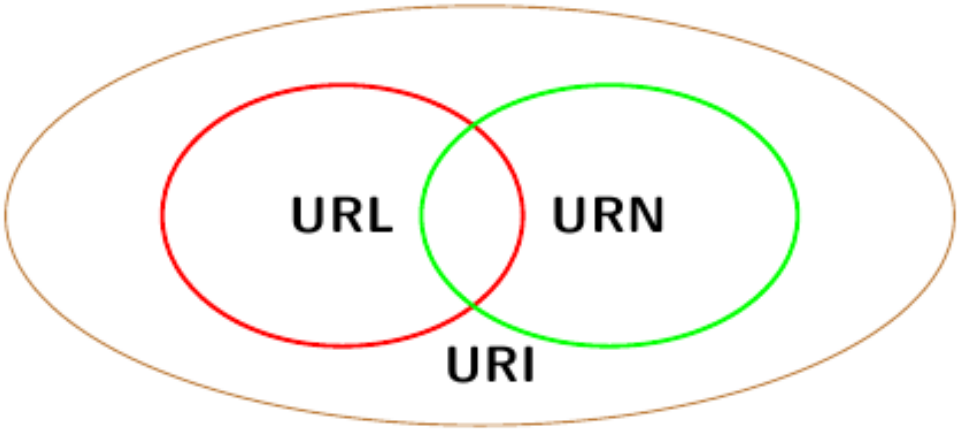


- A Uniform Resource Identifier (URI) is either a Uniform Resource Locator (URL) or a Uniform Resource Name (URN), or both
- It takes the form

Scheme:scheme-specific-part

- Conventions about the use of /, #, and ?

A note on URIs

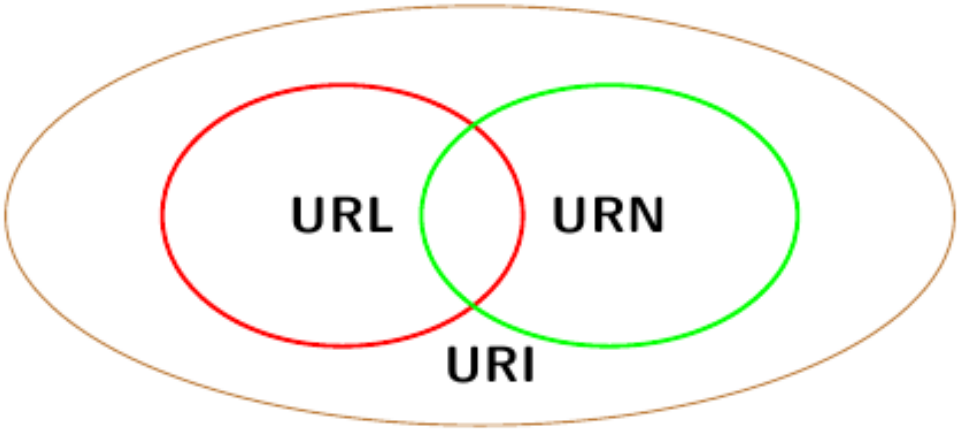


- A Uniform Resource Name functions like a person's name

urn:isbn:0-486-27557-4

That is: it defines an item's identity

A note on URIs

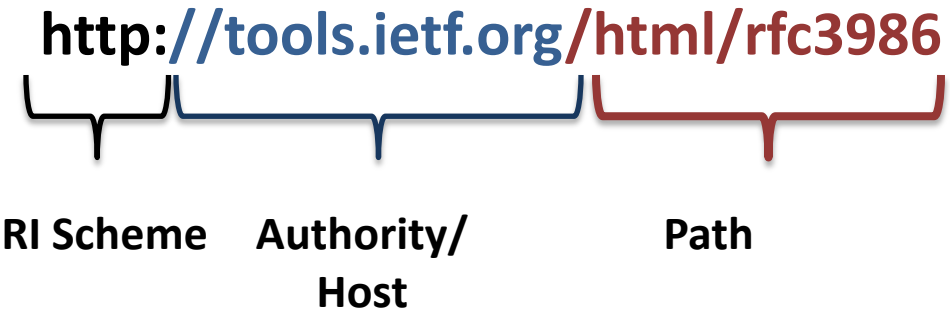


- A Uniform Resource Locator functions like a street address:

  http://www.google.com

  That is: it gives an item's location

The anatomy of a URL



Content negotiation

- Used to select “best” response for a request
  - Server-driven negotiation
  - Agent-driven negotiation
  - Transparent negotiation (not discussed here)

- Server-driven negotiation is normally used

Server-driven content negotiation

- User agent provides a list of preferences

```
Accept: text/html, …
Accept-language: us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-15,…
```

List of MIME media types

List of languages

List of content encodings

List of character sets

- Server decides “best” resource from those available
- Vary header may be returned to tell caches what request headers are/can be used to make the decision, e.g.,

```
Vary: Accept-language,Accept-Charset
```


Demo

Agent-driven content negotiation

- User agent provides a list of preferences, as before
- Server returns status code 300 and a list of choices (with their URIs) in either header fields or entity-body
- User-agent decides which one is “best” and issues new request for it (using the provided URIs)
- Requires multiple trips to the server
- Less widely used

Conditional requests

- Certain request headers can be included to make a request conditional
- Based on date and time of last modification:

```
If-Modified-Since: Sat, 28 Jan 2012 19:43:31 GMT
```

```
If-Unmodified-Since: Sat, 28 Jan 2012 19:43:31 GMT
```

- Based on entity tags (returned by server in ETAG: header)

```
If-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"
```

```
If-None-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"
```

- When specified condition is true, server returns requested resource, otherwise a status code is returned with no message-body (304 Not Modified or 412 Precondition Failed)

Design advantages of HTTP

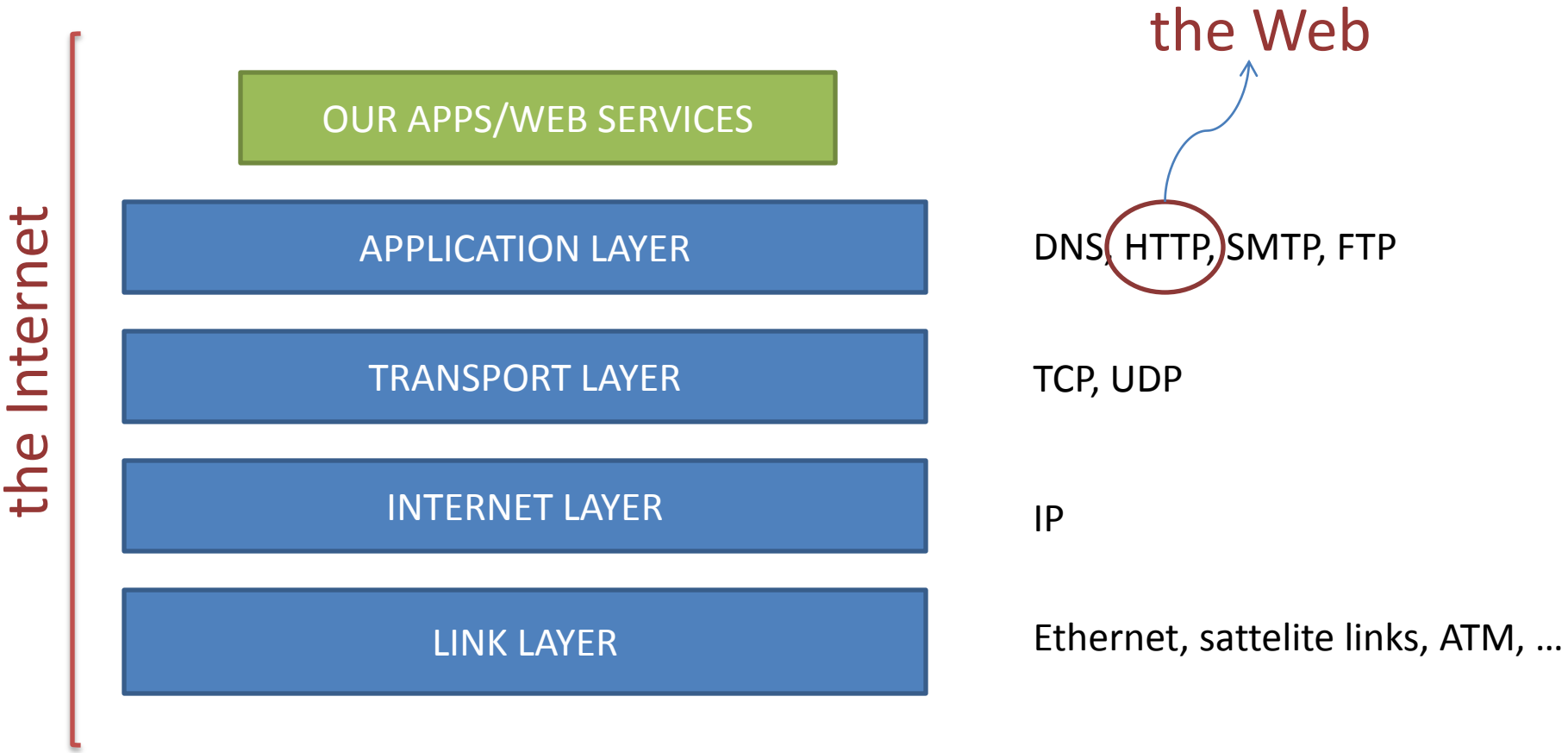
- Lightweight
- No client state on server, hence scalable (load balancing through multiple servers)
- HTTP brings a *uniform* interface to sharing data on the web (more on this later)

Web Servers

- A Web Server is a server that talks HTTP
- Responsibilities:
  1. Setup connection
  2. Receive & process HTTP requests
  3. Create & send HTTP response
  4. [Logging]

- Well-known web servers:
  - Apache HTTP Server [www.apache.org]
    Freely available
  - Microsoft Internet Information Services

Conclusion: The Web ≠ the Internet



SERVICES

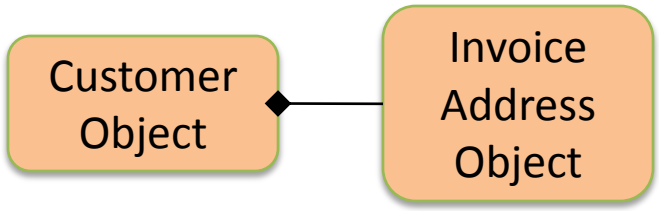
What is a service?

- The term **service** is like the term **multimedia**: lots of people have given different definitions.

- Essentially, a **service** is a software function or component:
  - It may carry out a business task,
  - provides access to files,
  - Perform generic functions like authentication and logging,
  - …

- Services reflect a new 'service-oriented' approach to programming, based on the idea of composing applications by discovering and invoking network-available services rather than building new applications or by invoking available applications to accomplish some task.

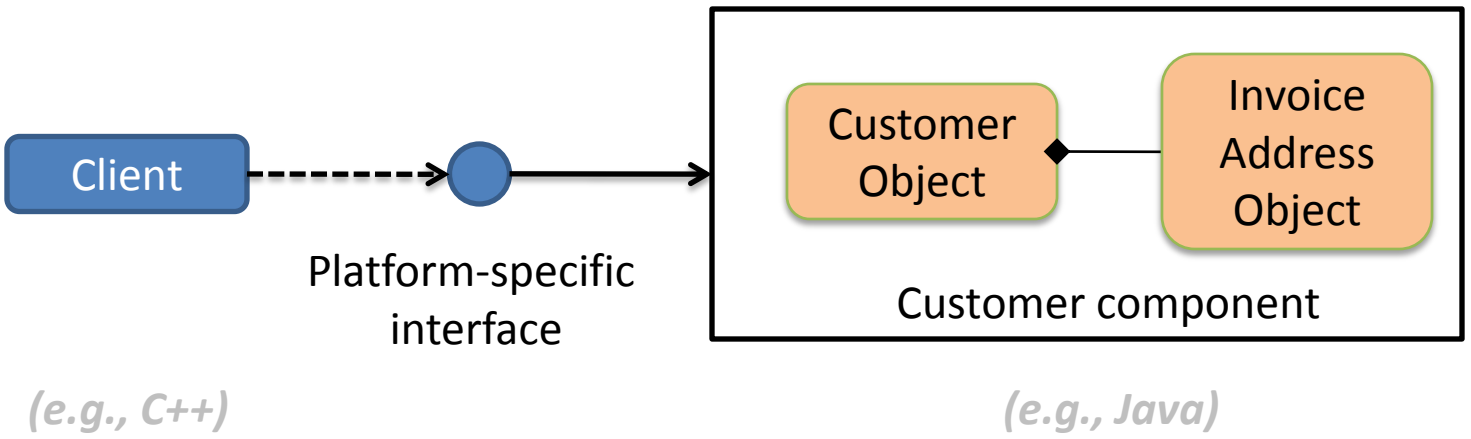
Some history: from local to distributed objects



(e.g., Java)

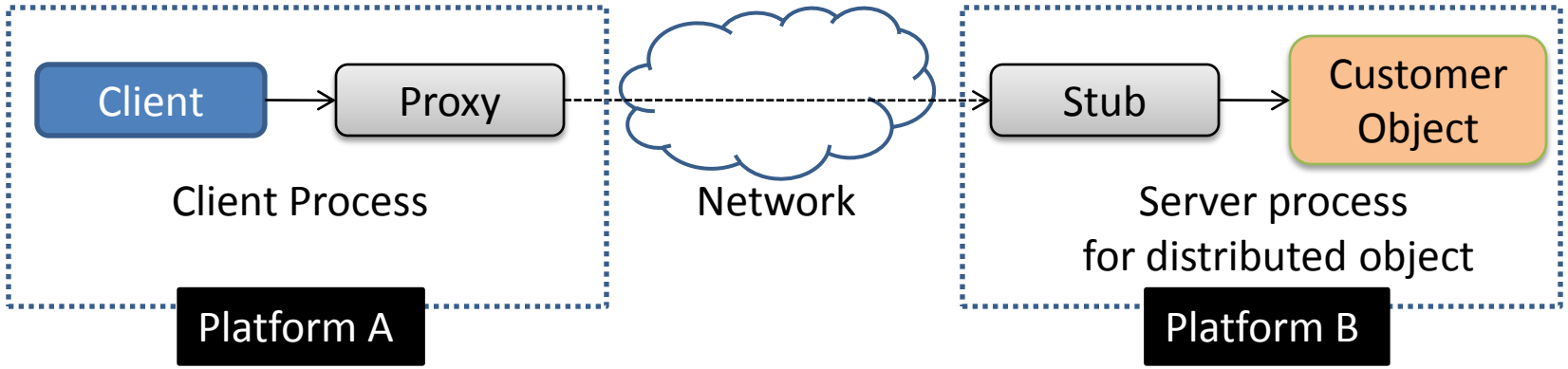
- **Object**-oriented programming allows encapsulation of both behavior and data.
- Clients use objects by first instantiating an object, and then calling their properties & methods
- Re-use is usually restricted to the same programming language and platform

Some history: from local to distributed objects



- **Components** were devised to facilitate software reuse across disparate programming languages.
- Components group related objects into (binary) units that can be plugged into applications (cf. electronic component assembly in circuit boards).
- Component reuse typically restricted to same computing platform due to incompatible binary interfaces.

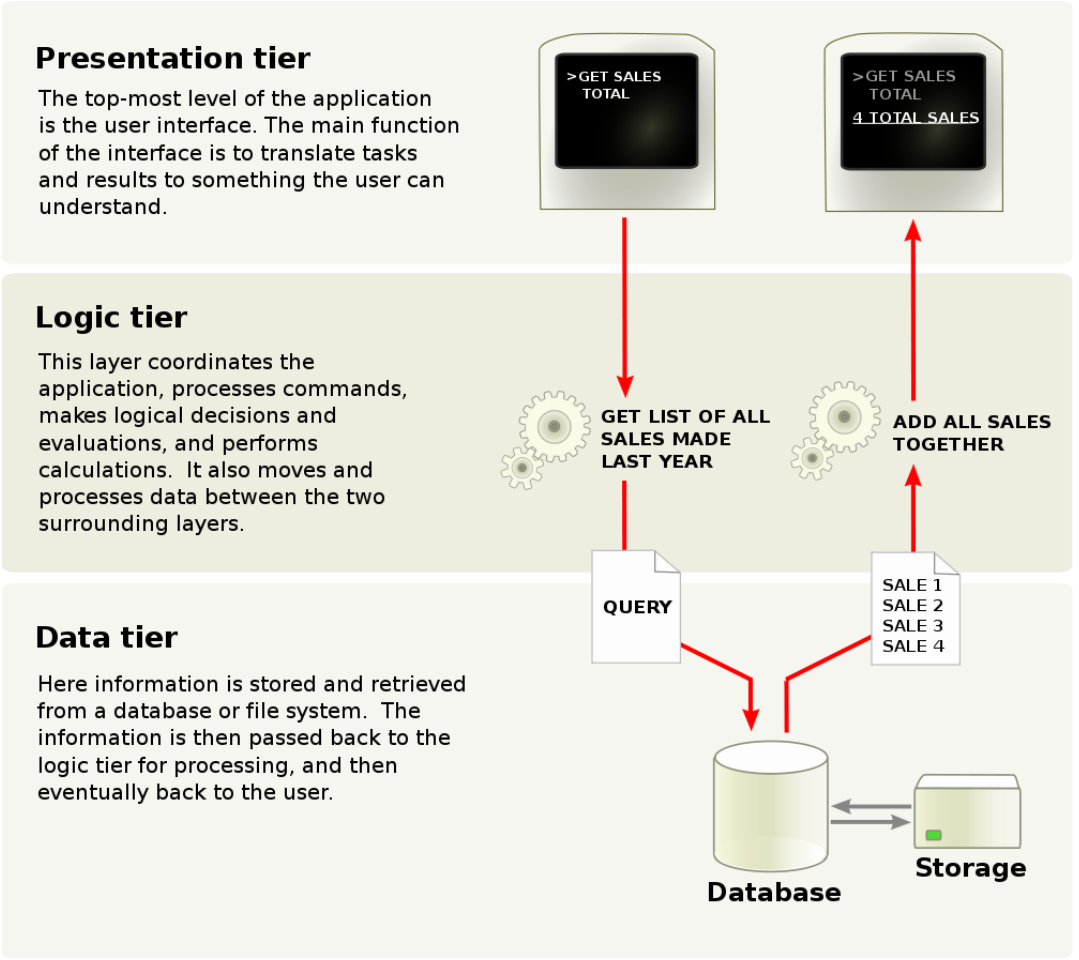
Some history: from local to distributed objects



- **Distributed objects.** To share and reuse objects, objects were deployed to remote servers. Clients connect to such objects through a remoting technology (CORBA, DCOM, Java or .NET Remote Method Invocation).
- Clients and distributed objects live in separate machines, and can therefore live in separate programming languages and platforms.
- Reuse restricted to the same remoting technology
- Scalability problems due to client state

Some history: from local to distributed objects

Here be services! →



- Distributed objects were quickly grouped into a logic tier that stores all of the application logic – these are already "services"

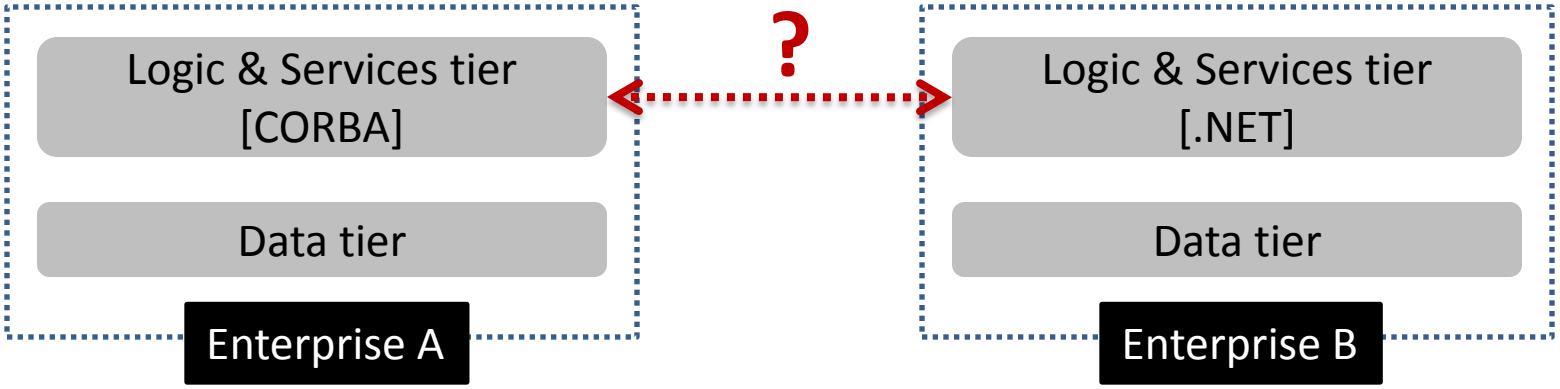
What is a service? (cont.)

- Services hence provide logical functions that are shared across different applications. They enable software that runs on disparate computing platforms to collaborate.

- A platform may be any combination of hardware, operating systems (e.g., Windows, Linux, Android, iOS), software framework (java, .Net, Rails), and programming language.

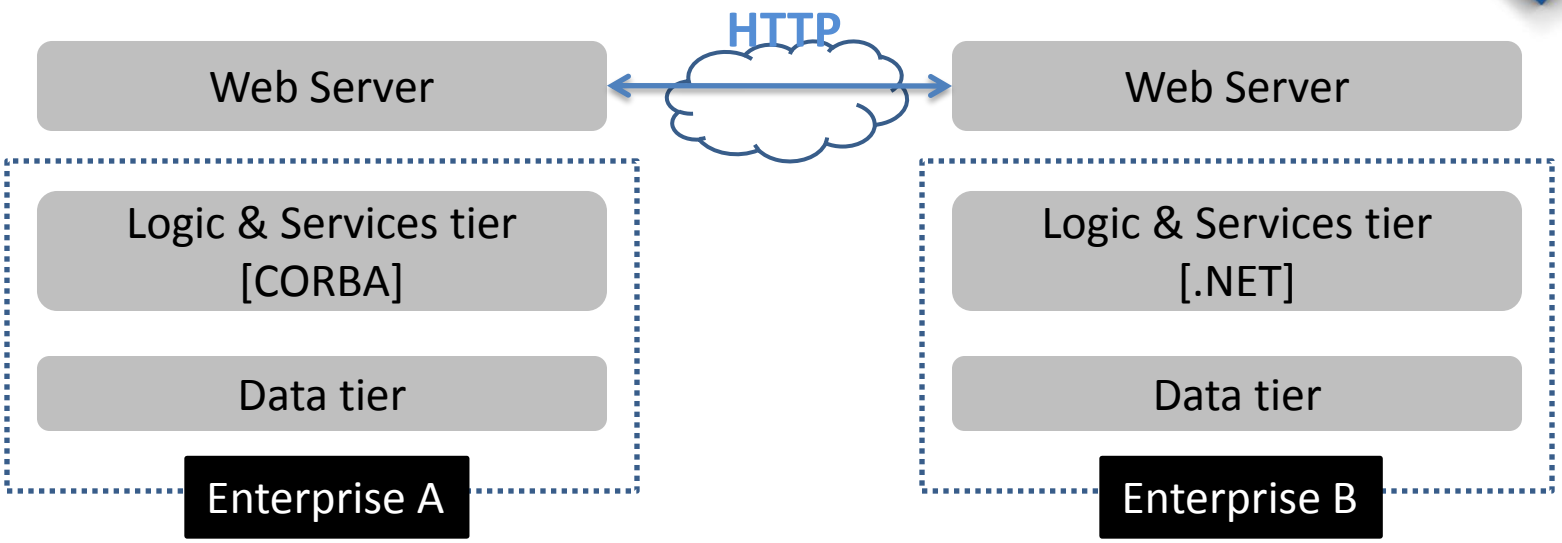
- The service-oriented paradigm to programming utilizes services as the constructs to support the rapid development of easily composable distributed applications (again cf. electronic component assembly in circuit boards).

What is a Web Service?



- Service reuse is restricted to the same remoting technology when built on traditional distributed object architectures.
- This is especially problematic in enterprise integration and communication scenarios, where services must be callable from outside enterprise boundaries.

What is a Web Service?



- **Web services** integrate disparate systems and expose reusable business functions over the web (HTTP).
- They leverage HTTP either:
  - as a simple transport over which data is carried (e.g., SOAP/WSDL services), or
  - a complete pre-defined application protocol (RESTful services).

Where are Services used?

- Within an enterprise (Enterprise Application Integration)
  - Accelerate and reduce the cost of integration
  - Save on infrastructure deployment and management costs
  - Reduce skill requirements
  - Improve reuse

- Between enterprises (E-business integration, B2B)
  - Providing service to a company's customers
  - e.g., an Insurance company wishes to link its systems to the systems of a new institutional customer

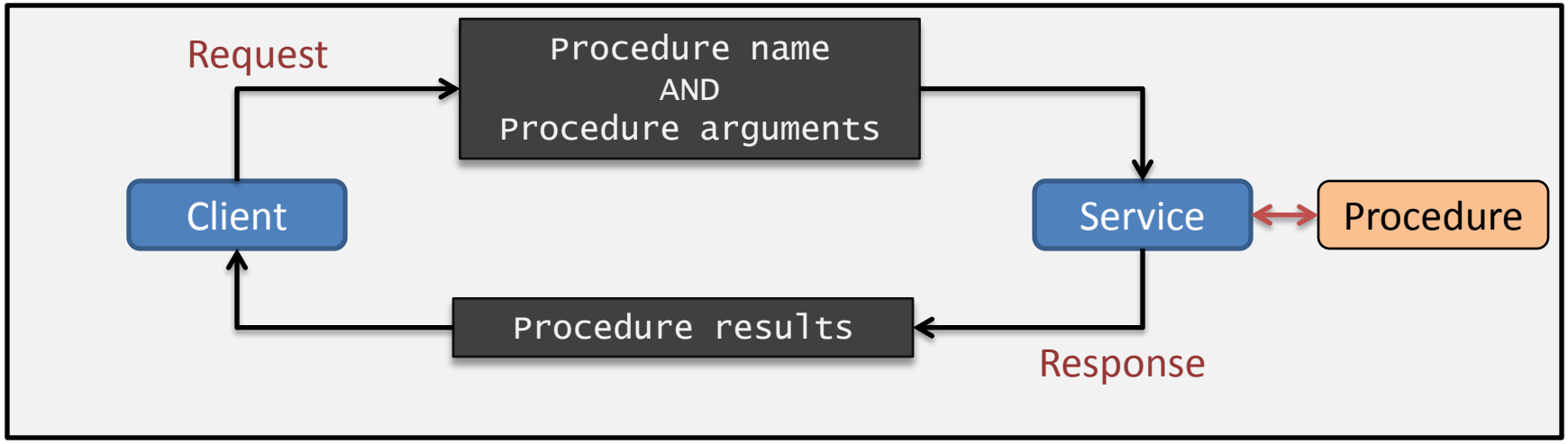
  - Accessing services from a company's partners and suppliers
  - e.g., dynamically link to new partners and suppliers to offer their services to complement the value the company provides

  - Standards and common infrastructure reduce the barriers
  - Simplicity accelerates deployment
  - Dynamics opens new business opportunities

Web Service API styles

- RPC (Remote Procedure Call)
- Message-based
- Resource-based

RPC Style (1/2)



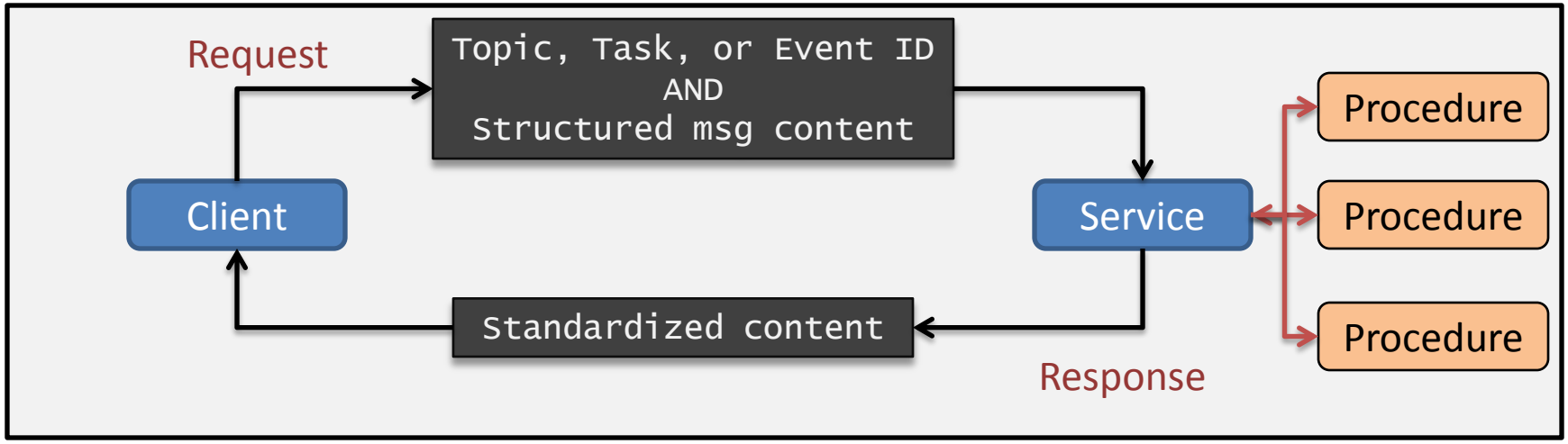
- Client sends message to a remote server and blocks while waiting for response
- Request message identifies the procedure to be executed and its arguments
- Server decodes message, maps message arguments directly to input parameters, executes procedure, and sends (serialized) results back to client

RPC Style (2/2)

- Pros:
  - Very easy to implement (lots of frameworks that automate the process, e.g. AX-WS framework for Java)

- Cons:
  - Usually inflexible and fragile: tight coupling between client and service, if procedure needs to change (e.g., number of arguments), all clients need to be rewritten.
  - Usually restricted to **synchronous communication** (client blocks while waiting for response)

Message-based style



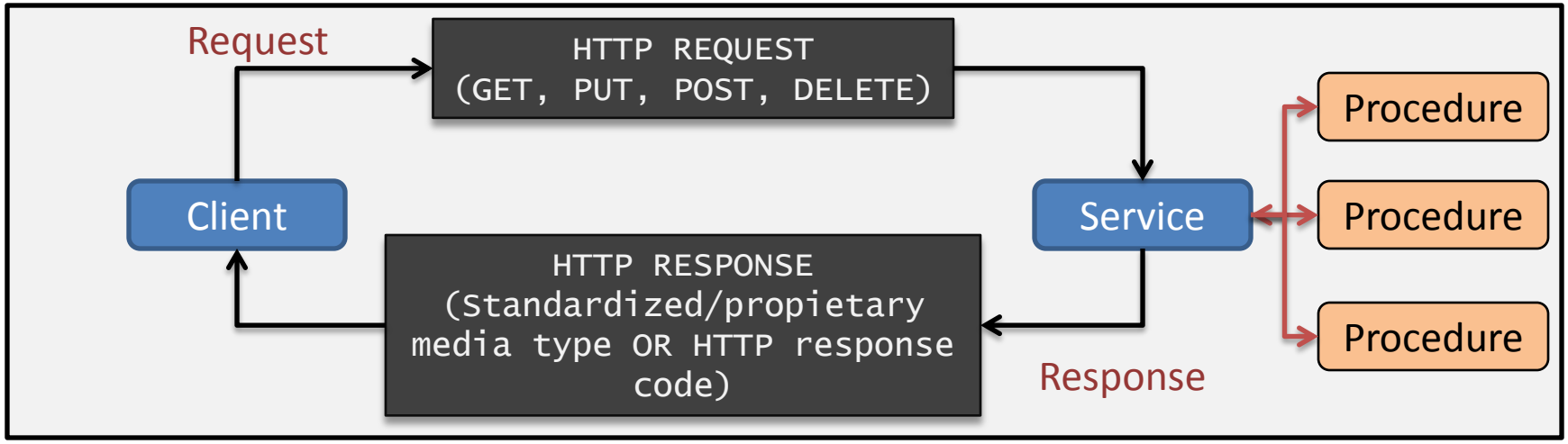
- In a message-based API, messages are not derived from the signatures of remote procedures.
- Instead, messages may carry information on specific topics, tasks to execute, and events.
- The server selects the correct procedure to execute based on the message content

Message-based Style (2/2)

- Pros:
  - Looser coupling between clients and servers
  - Support for asynchronous communication [necessary on web-scale networks]

- Cons:
  - Messages must be standardized somehow. This is easy if communication is within the same organization, but more difficult when many parties are involved.

Resource-based style



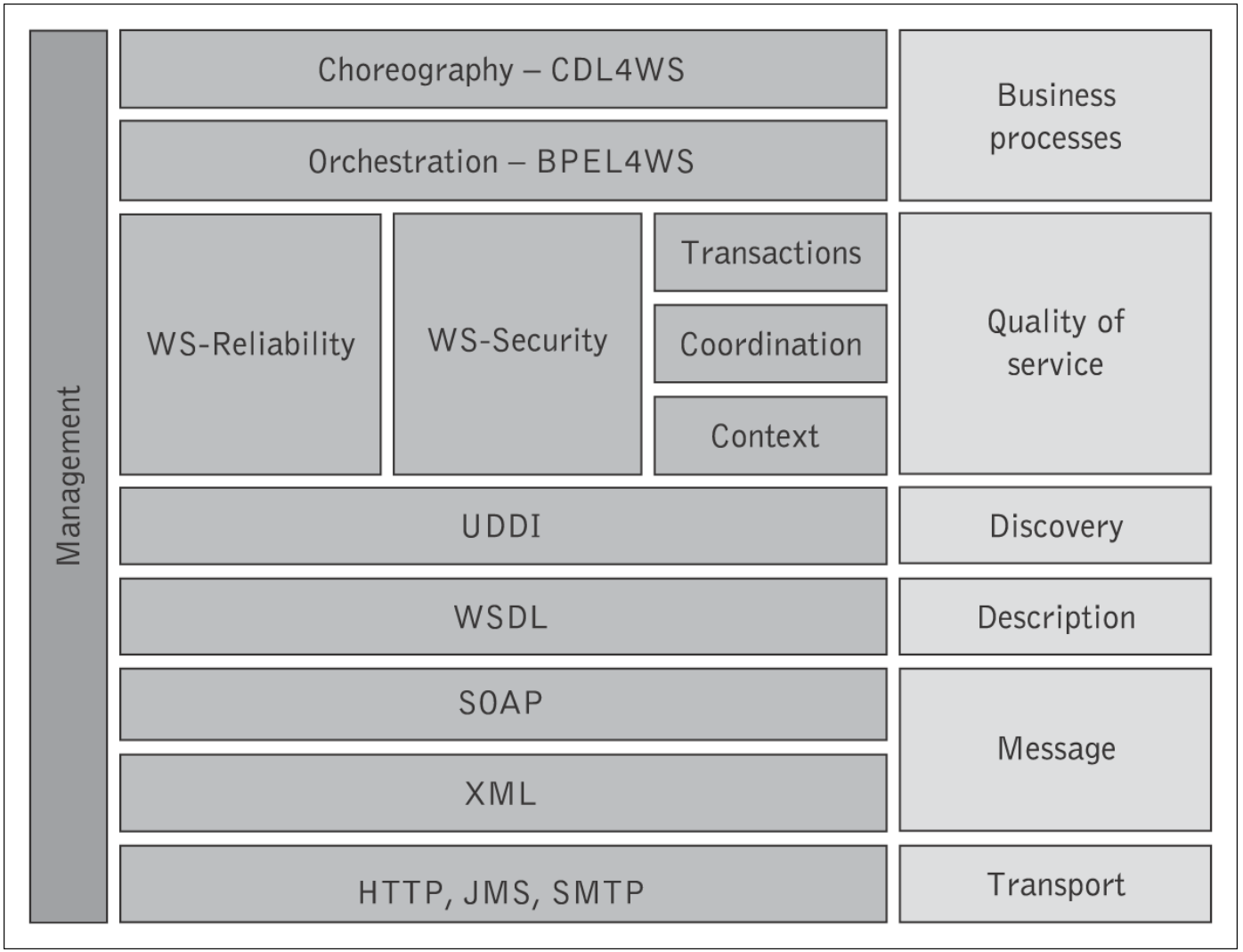
- In a resource-based APSI, all procedures, instances of domain data, and files are given a URI.
- HTTP is used as a complete application protocol to define standard service behavior.
- Information is exchanges based on standardized media types (JSON, XML, ATOM, …) and HTTP response codes where possible
- Clients manipulate the state of resources through representations (e.g., a database table row may be represented as XHTML, XML, or JSON).

Two competing technology stacks

- **Big Web Services (WS-*)**
  - Various (complex) protocols on top of HTTP (SOAP, UDDI, WSDL, WS-Addressing, …)
  - Is mostly used to implement RPC-style services, but can be used to implement any of the three
  - Lots of standards! Primarily meant to create web services that involve more than 2 peers.

- **RESTful Web Services**
  - Use ONLY HTTP and standar media types
  - Restricted to Resouce-style services
  - Conceptually simpler, but mainly restricted to web services that are limited to two endpoints

WS-* Technology Stack



Web Services Standards Overview

Interoperability Issues

- Basic Profile 1.1 — WS-I — Final Specification
- Basic Profile 1.2 — WS-I — Working Group Draft
- Basic Profile 2.0 — WS-I — Working Group Draft
- Attachments Profile 1.0 — WS-I — Final Specification
- Simple SOAP Binding Profile 1.0 — WS-I — Final Specification
- Basic Security Profile 1.0 — WS-I — Board Approval Draft
- REL Token Profile 1.0 — WS-I — Working Group Draft
- SAML Token Profile 1.0 — WS-I — Working Group Draft
- Conformance Claim Attachment Mechanism (CCAM) 1.0 — WS-I — Final Specification
- Reliable Asynchronous Messaging Profile (RAMP) 1.0 — WS-I — Working Draft

Business Process Specifications

- Business Process Execution Language for Web Services 1.1 (BPEL4WS) — 1.1 — BEA Systems, IBM, Microsoft, SAP, Siebel Systems — OASIS-Standard
- WS-Choreography Model Overview — 1.0 — W3C — Working Draft
- Web Service Choreography Interface (WSCI) — 1.0 — W3C — Sun Microsystems, SAP, BEA Systems and Intalio — Note
- Web Service Choreography Description Language (CDL4WS) — 1.0 — W3C — Candidate Recommendation
- Business Process Execution Language for Web Services 2.0 (BPEL4WS) — 2.0 — OASIS, BEA Systems, IBM, Microsoft, SAP, Siebel Systems — Committee Draft
- Business Process Management Language (BPML) — 1.0 — BPMI.org — Final Draft
- XML Process Definition Language (XPDL) — 2.0 — Final

Management Specifications

- Management Using Web Services (WSDM-MUWS) — 1.0 — OASIS — OASIS-Standard
- Management Of Web Services (WSDM-MOWS) — 1.0 — OASIS — OASIS-Standard
- WS-Management — AMD, Dell, Intel, Microsoft and Sun Microsystems — Published Specification
- Service Modeling Language — IBM, BEA, BMC, Cisco, Dell, HP, Intel, Microsoft, Sun — Draft Specification

Presentation Specifications

- Web Services for Remote Portlets (WSRP) — 2.0 — OASIS — Committee Draft

Metadata Specifications

- WS-Policy — 1.5 — W3C — Working Draft
- WS-PolicyAssertions — 1.1 — BEA Systems, IBM, Microsoft, SAP — Public Draft
- WS-PolicyAttachment — 1.2 — W3C — W3C Member Submission
- WS-Discovery — Microsoft, BEA Systems, Canon, Intel and webMethods — Draft
- WS-MetadataExchange — 1.1 — BEA Systems, Computer Associates, IBM, Microsoft, SAP, Sun Microsystems and webMethods — Public Draft
- Universal Description, Discovery and Integration (UDDI) — 3.0.2 — OASIS — OASIS-Standard
- Web Service Description Language 2.0 SOAP Binding — 2.0 — W3C — Working Draft
- Web Service Description Language 2.0 Core — 2.0 — W3C — Candidate Recommendation
- Web Service Description Language 1.1 — 1.1 — W3C — Note

Reliability Specifications

- WS-ReliableMessaging — 1.1 — OASIS — Committee Draft
- WS-Reliable Messaging Policy Assertion (WS-RM Policy) — 1.1 — OASIS — Committee Draft
- WS-Reliability — 1.1 — OASIS — OASIS-Standard

Security Specifications

- WS-Security — 1.1 — OASIS — OASIS-Standard
- WS-SecurityPolicy — 1.1 — IBM, Microsoft, RSA Security, VeriSign — Public Draft
- WS-Security: SOAP Message Security — 1.1 — OASIS — Public Review Draft
- WS-Security: Username Token Profile — 1.1 — OASIS — Public Review Draft
- WS-Security: Kerberos Binding — 1.0 — Microsoft, IBM, OASIS — Working Draft
- WS-Federation — 1.1 — IBM, Microsoft, BEA Systems, RSA Security, and VeriSign — Initial Draft
- WS-Security: SAML Token Profile — 1.1 — OASIS — Public Review Draft
- WS-Trust — BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology — Initial Draft
- WS-Security: X.509 Certificate Token Profile — 1.1 — OASIS — Public Review Draft
- WS-SecureConversation — BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology — Public Draft

Transaction Specifications

- WS-Coordination — 1.1 — OASIS — Working Draft
- WS-Business Activity — 1.1 — OASIS — Working Draft
- WS-Atomic Transaction — 1.1 — OASIS — Committee Draft
- WS-Composite Application Framework (WS-CAF) — 1.0 — Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems — Committee Specification
- WS-Context (WS-CTX) — 1.0 — Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems — Committee Draft
- WS-Coordination Framework (WS-CF) — 1.0 — Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems — Committee Draft
- WS-Transaction Management (WS-TXM) — 1.0 — Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems — Committee Draft

Resource Specifications

- Web Services Resource Framework (WSRF) — 1.2 — OASIS — OASIS-Standard
- WS-BaseFaults (WSRF) — 1.2 — OASIS — Working Draft
- WS-ServiceGroup (WSRF) — 1.2 — OASIS — Working Draft
- WS-ResourceProperties — 1.2 — OASIS — Working Draft
- WS-ResourceLifetime — 1.2 — OASIS — Working Draft
- WS-Transfer — W3C — W3C Member Submission
- Resource Representation SOAP Header Block (RRSHB) — W3C — Recommendation

Messaging Specifications

- WS-Notification — 1.3 — OASIS — OASIS-Standard
- WS-BrokeredNotification — 1.3 — OASIS — OASIS-Standard
- WS-BaseNotification — 1.3 — OASIS — OASIS-Standard
- WS-Eventing — W3C — Public Draft
- WS-Addressing – Core — 1.0 — W3C — Recommendation
- WS-Enumeration — Systinet, Microsoft, Sonic Software, BEA Systems and Computer Associates — Public Draft
- WS-Topics — 1.3 — OASIS — OASIS-Standard
- WS-Addressing – WSDL Binding — 1.0 — W3C — Candidate Recommendation
- WS-Addressing – SOAP Binding — 1.0 — W3C — Recommendation

SOAP

- SOAP — 1.2 — W3C — Recommendation
- SOAP Message Transmission Optimization Mechanism (MTOM) — 1.0 — W3C — Recommendation
- SOAP — 1.1 — W3C — Note

XML Specifications

- XML 1.1 — 1.1 — W3C — Recommendation
- XML 1.0 — 1.0 — W3C — Recommendation
- Namespaces in XML — 1.1 — W3C — Recommendation
- XML Information Set — 1.0 — W3C — Recommendation
- XML Schema — 1.1 — W3C — Working Draft
- XML binary Optimized Packaging (XOP) — 1.0 — W3C — Recommendation
- Describing Media Content of Binary Data in XML (DMCBDX) — W3C — Note

Standards Bodies

- OASIS — The Organization for the Advancement of Structured Information Standards (OASIS)
- W3C — The World Wide Web Consortium (W3C)
- WS-I — The Web Services Interoperability Organization (WS-I)
- IETF — The Internet Engineering Task Force (IETF)

Dependencies

Messaging Specifications · Metadata Specifications · Security Specifications · Reliability Specifications · Resource Specifications · Management Specifications · Business Process Specifications · Transaction Specifications · Presentation Specifications

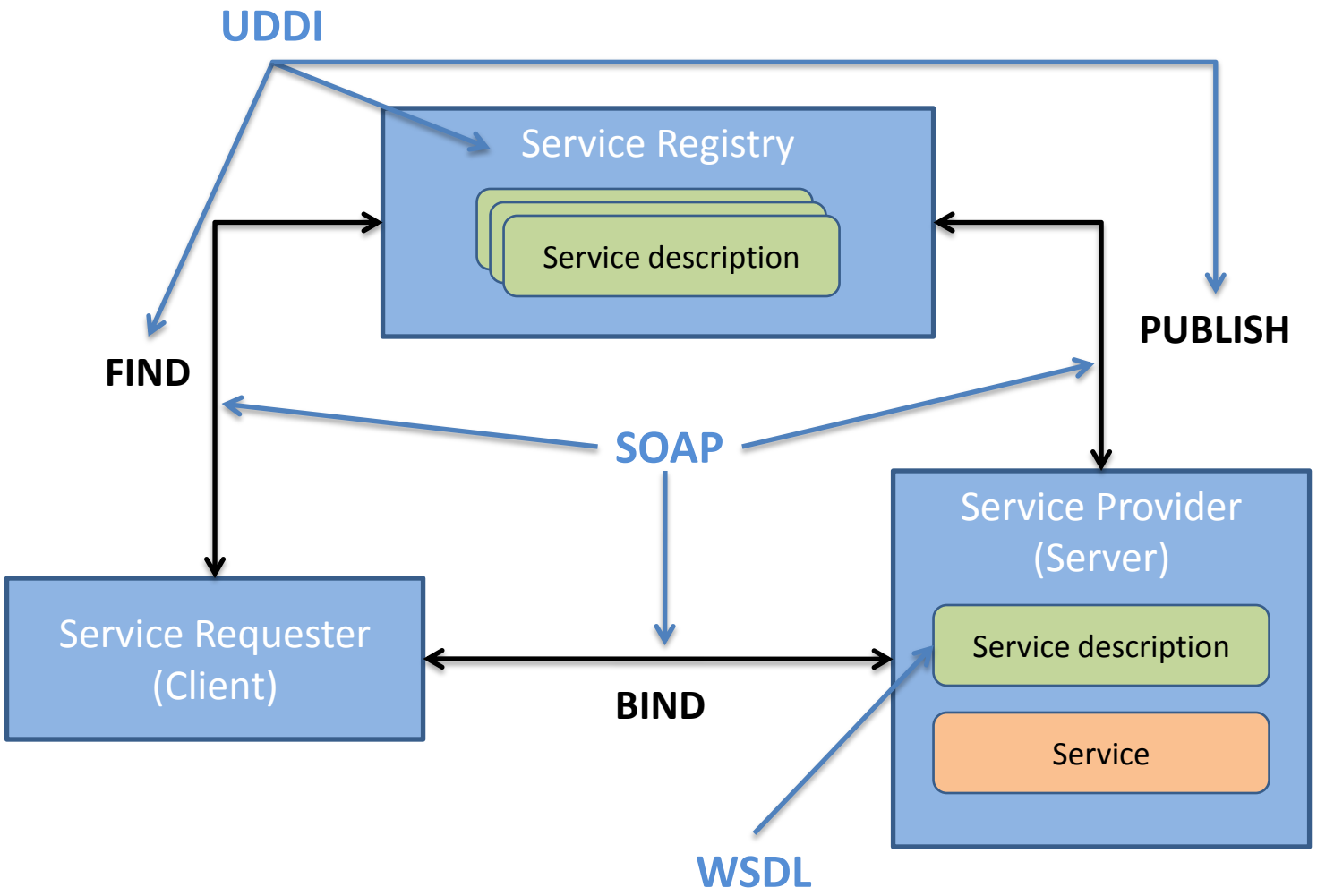
Version 3.0 · February 2007

innoQ

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 21 02 77 162-100
info@innoq.com · www.innoq.com

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

WS-* Message, Description & Discovery



- While SOAP & WSDL are frequently used, UDDI has never caught on.

A word of caution

- Web Service call entail distributed communication & programming
  - Network latency
  - Failures
  - Complexity of distributed programming

- So using web services only makes sense in situations where out-of-processes and cross-machine calls make sense.

References

- R. Daignau, Service Design Patterns, Addisson-Wesley
- M. P. Papazoglou, Web Services: Principles and Technology, Prentice Hall