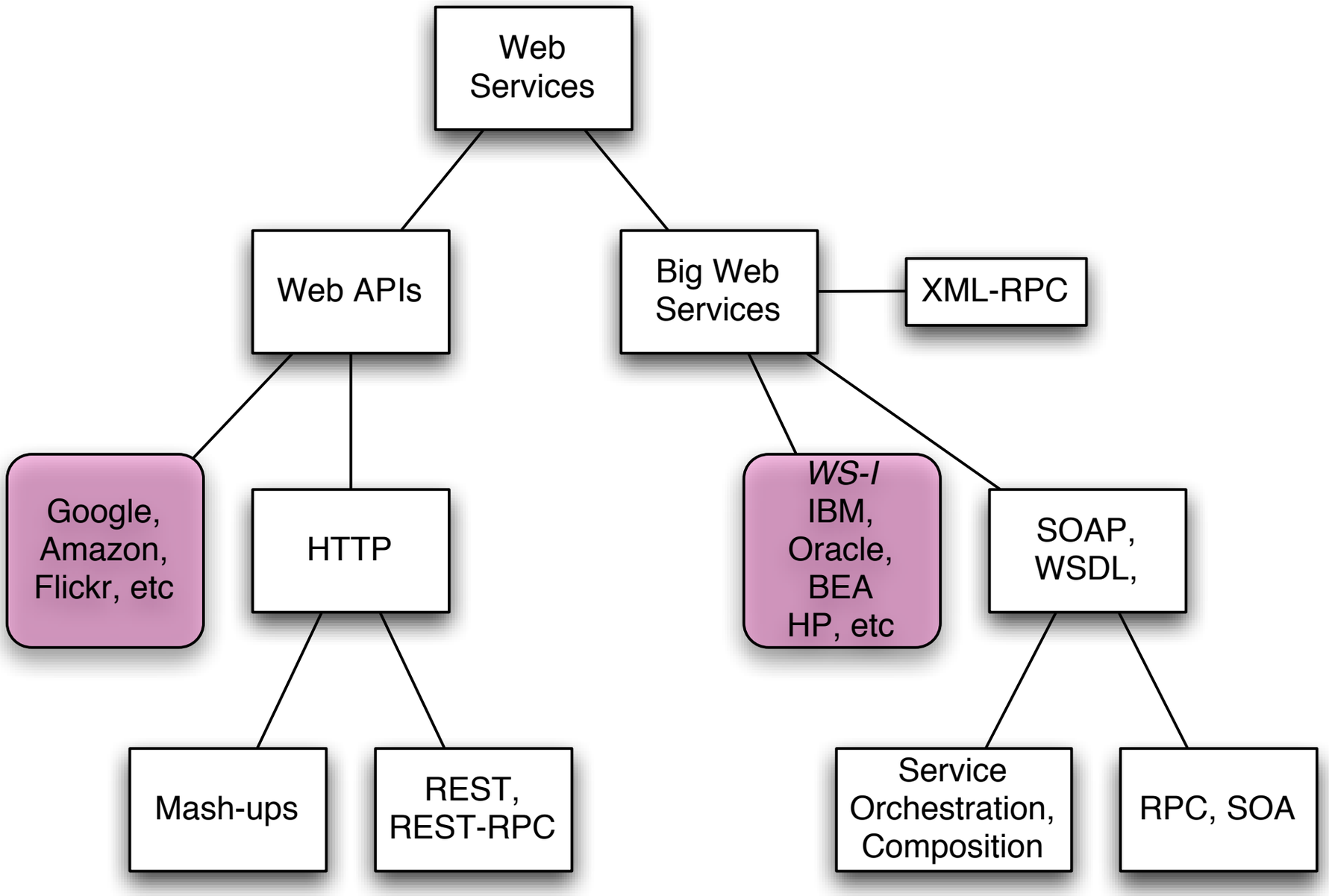
Multi-agent and Semantic Web Systems: The Programmable Web

Fiona McNeill

School of Informatics

4th March 2013

WS: High-level view



Two perspectives

- Web APIs
  - Same technology that supports existing WWW also supports web services
  - If its on the Web, it's a web service — Richardson and Ruby (2007), *RESTful* Web Services, O'Reilly.
- Big Web Services:
  - WS interfaces are specified in WSDL
  - WS exchange data in SOAP messages

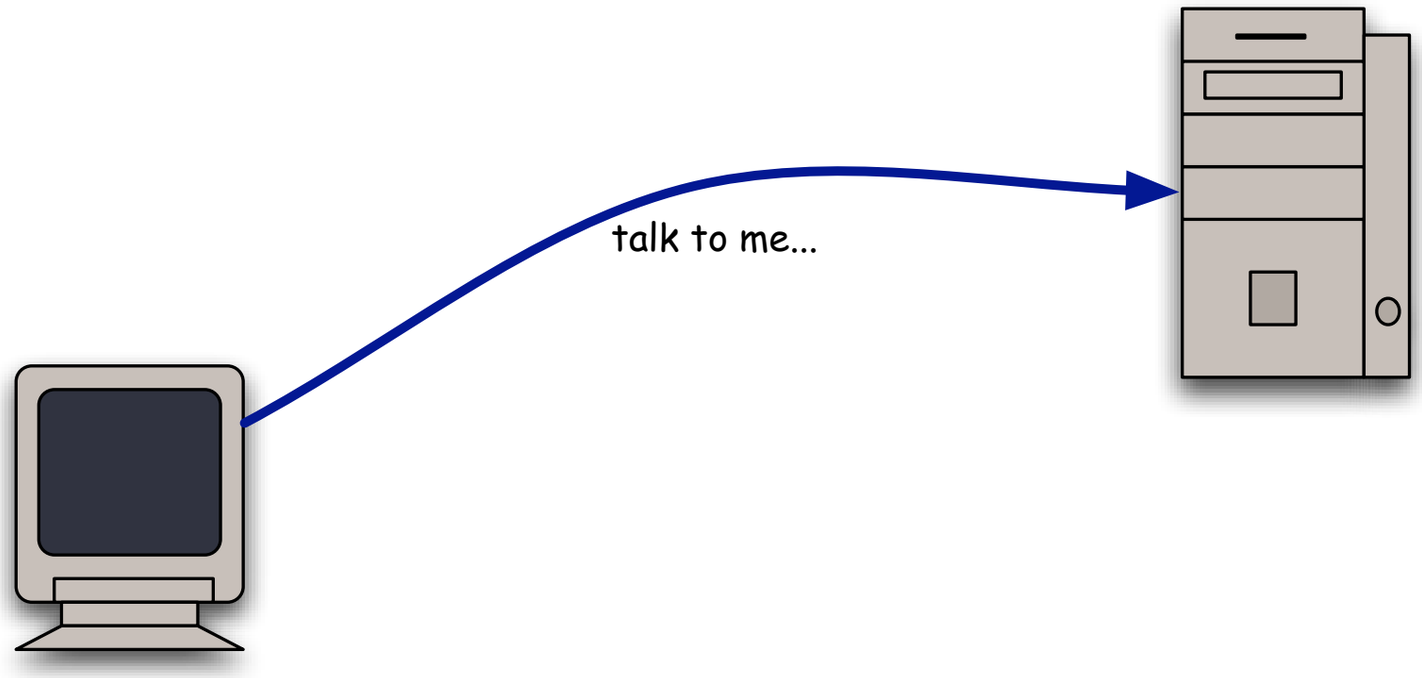
WS: Computational Styles / Architecture

1. Representational State Transfer (REST) – only uses HTTP methods and resources (i.e., URIs)

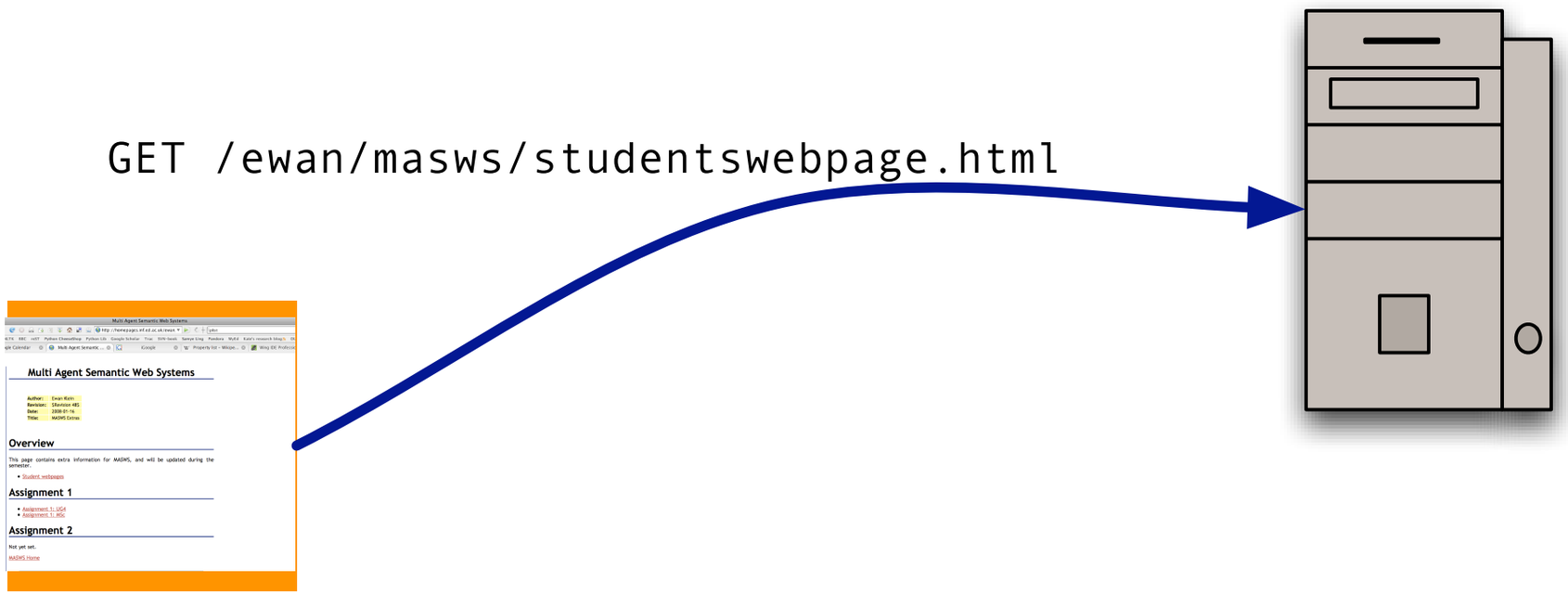
2. Remote procedure call (RPC) — distributed programming paradigm

3. Service Orient Architecture (SOA) — emphasis on messages and metadata for service functionality

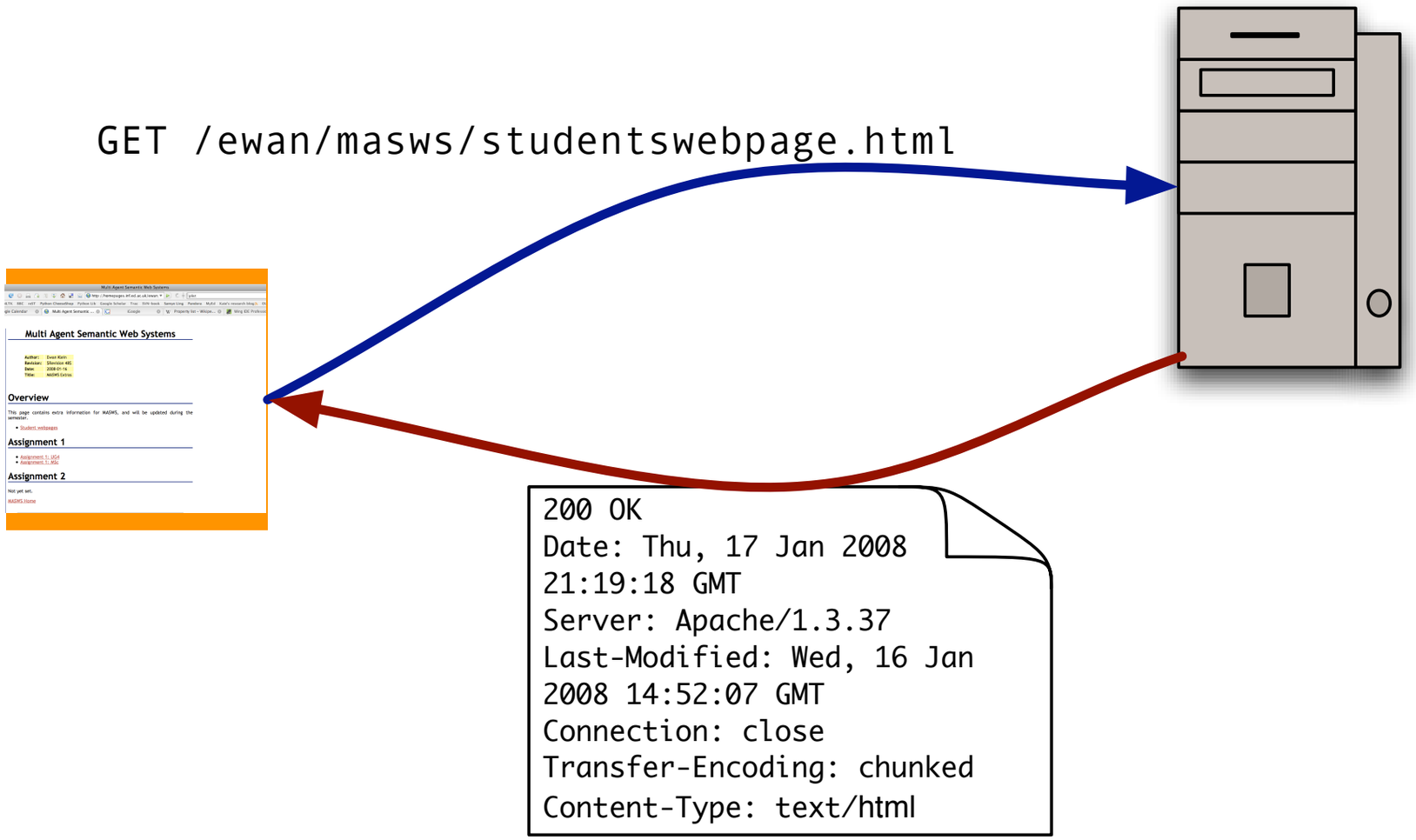
HTTP Client-Server



HTTP Browser-Server



HTTP Browser-Server



HTTP Client-Server Interaction, 1

- Client opens TCP/IP socket connection on port 80 to host.

HTTP Client-Server Interaction, 1

- Client opens TCP/IP socket connection on port 80 to host.
- Client sends an HTTP request consisting of a method and request URL.

HTTP Client-Server Interaction, 1

- Client opens TCP/IP socket connection on port 80 to host.
- Client sends an HTTP request consisting of a method and request URL.
- Server sends back a message consisting of a status code and some header information, followed by a blank line.

HTTP Client-Server Interaction, 1

- Client opens TCP/IP socket connection on port 80 to host.

- Client sends an HTTP request consisting of a method and request URL.

- Server sends back a message consisting of a status code and some header information, followed by a blank line.

- If the client's request method is a GET, and the server's status code is 200 OK, then server also returns a representation of the requested resource.

HTTP Client-Server Interaction, 2

HTTP Request

```
HOST: homepages.inf.ed.ac.uk
GET /ewan/masws/studentswebpage.html
```

METHOD + argument

Client-Server Interaction, 3

Response Status Code

```
200 OK
```

Client-Server Interaction, 4

Server Headers

```
Date: Thu, 17 Jan 2008 21:19:18 GMT
Server: Apache/1.3.37
Last-Modified: Wed, 16 Jan 2008 14:52:07 GMT
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
```

Client-Server Interaction, 5

Response Document

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<head>
  <title>MASWS Student Webpages</title>
  <link href="delpost.css" rel="stylesheet" type="text/css">
  </head>
<body>
  <h1>MASWS Student Webpages</h1>
  ...
```

What's in the document?

- HTML — for rendering by browser
- XML — anything that can deal with structured data
- JSON — 'lightweight' alternative to XML for data serialization

audio, graphics, etc ...

- MIME types (e.g., `text/html`)

JSON

Javascript Object Notation

- http://www.json.org/

- Language-independent scheme for exchanging data between applications.
  - lightweight format (i.e., compared to XML)
  - easy for humans to read and write
  - easy for machines to parse and generate

- Two basic structures:
  1. unordered key/value pairs
  2. ordered list of values
- A similar non-XML data format language: YAML http://www.yaml.org/

JSON Examples

Dictionary and List

```
{"HOST": "homepages.inf.ed.ac.uk",
 "PATH": "/ewan/masws/index.html"}

["homepages.inf.ed.ac.uk", "del.icio.us"]
```

JSON Examples

Dictionary and List

```
{"HOST": "homepages.inf.ed.ac.uk",
 "PATH": "/ewan/masws/index.html"}

["homepages.inf.ed.ac.uk", "del.icio.us"]
```

Dictionary with list values

```
{"HOSTS": ["inf.ed.ac.uk", "del.icio.us"],
 "PATHS": []}
```

JSON Examples

Dictionary and List

```
{"HOST": "homepages.inf.ed.ac.uk",
 "PATH": "/ewan/masws/index.html"}

["homepages.inf.ed.ac.uk", "del.icio.us"]
```

Dictionary with list values

```
{"HOSTS": ["inf.ed.ac.uk", "del.icio.us"],
 "PATHS": []}
```

List of dictionaries

```
[{"HOST": "inf.ed.ac.uk"}, {"HOST": "del.icio.us"}]
```

A GET Request in Python

Parsing a URL

```
>>> import httplib
>>> h = httplib.HTTPConnection('homepages.inf.ed.ac.uk')
>>> h.request('GET', '/ewan/masws/studentswebpage.html')
>>> r = h.getresponse()
>>> print r.status, r.reason
200 OK
>>> for m in r.msg.headers: print m,
...
Date: Fri, 18 Jan 2008 09:51:50 GMT
Server: Apache/1.3.37
Last-Modified: Wed, 16 Jan 2008 14:52:07 GMT
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
```

HTTP Methods

GET	Requests a representation of the specified resource.
HEAD	Asks for the response identical to the one that would correspond to a GET request, but without the response body.
POST	Submits data to be processed (e.g. from an HTML form) to the identified resource.
PUT	Uploads a representation of the specified resource.
DELETE	Deletes the specified resource.

HTTP Status Codes

1xx Informational

2xx Success

E.g., `200 OK`

3xx Redirection

E.g., `303 See Other`

4xx Client Error

E.g., `404 Not Found`

5xx Server Error

Web API Summary

- Opening a browser and typing in a URL initiates a kind of client-server interaction.
- Client program sends a request to a host, server sends a response. HTTP provides a kind of envelope for messages.
- Server response depends in part on the HTTP method; may also be encoded in the URL.
- Issues about RPC approach will be looked at later in course.
- Returned document can be in a variety of formats.
- XML and JSON: both examples of data-exchange formats.
- `http://www.programmableweb.com/` for APIs and mashups