Assignment 2 Aftermath
Distributed Systems HS 2010
Simon Mayer
Today's Menu

- Results Assignment 2
- Hints (REST vs. WS-*)
- Submission Considerations + Example Solution
- Briefly: Sockets
Results

- Average grade of **5.33**
- Reports much improved
- Grade 6.0: some nice enhancements
REST and WS-*

- Summarize REST:
  - Every resource is identified with a unique URI
  - Uniform Interface (e.g., HTTP verbs GET, POST, PUT, DELETE, . . .)
  - Multiple representations or resources: (specified by setting the Accept header)
  - Stateless communication
Take-Home Point: REST

- REST defines a number of constraints. Thus, when selecting a RESTful architectural style, you constrain your application to:
  - Using only the uniform interface (e.g., only HTTP verbs)
  - Keeping no client context on the server
  - Adopting a Resource-Oriented Architecture with URIs
  - ...

REST and WS-*

- Summarize WS-*:
  - Uses HTTP as transport protocol (as opposed to application-level)
  - Describes the service through an WSDL file
  - Messages are exchanged in SOAP format, and encapsulated in an XML envelope
REST and WS-*

- Advantages of REST vs. WS-*:
  - Lightweight
  - Human readable, easy debugging
  - Widely available tools, e.g. Web browser, HTTP libraries

- Advantages of WS-* vs. REST:
  - Agreed format between client and server: WSDL as strict specification
  - (Usually simple to consume -> M2M)
  - Strong typing
REST and WS-*

- Which one to use?
  - It depends on the specific application to be deployed...
  - Is my application Resource-Oriented or Service-Oriented?

- Decide whether it pays off to map your application to a ROA:
  - Yes  Using REST could be beneficial...
  - No   Use WS-*
Resource-Oriented Web Application Example

- Amazon Web Services Simple Storage Service (S3)
  - S3 stores arbitrary objects up to 5 gigabytes in size
  - Objects are organized into buckets (each owned by an account)

GET http://s3.amazonaws.com/bucket/key maps smoothly to “Retrieve bucket”
DELETE http://s3.amazonaws.com/bucket/key maps smoothly to “Delete bucket”
PUT http://s3.amazonaws.com/bucket/key maps smoothly to “Create new bucket”
POST http://s3.amazonaws.com/bucket/key maps smoothly to “Modify data in bucket”
Submission Considerations...

- Make sure your project compiles and runs...

- Include all libraries, use relative paths and not "C:/simons_folder"

- Task 5: Specify the port number used and give an overview of the Web interface of your server

- Make your interfaces useable (button names should be descriptive, progress bars should state the current value and unit)
Figures in Reports

- Should not only be there, but also useful....

- Generic -->

- Pointless -->
Figures in Reports

- Good... ! And, in this case, very helpful to understand the architecture
6.0 Grade Example

- All tasks completed

- Excellent reports (+ online ReportComments)
  - Insight into implementation specifics, not general issues
  - Make use of diagrams! They are very useful to convey ideas.
  - Use a professional writing style: imagine showing the report to a business partner.

- Enhancements:
  - Multi-threaded server, SOAP by hand, Visualization by hand
  - Nice UIs, extra sensors and sophisticated functionality
6.0 Grade Example

- Task 4: Cloud visualization
  - Values update automatically
  - Implemented also locally (using a library)
6.0 Grade Example

- Task 5: Web-browser interface
  - Extra sensor/functionality
  - Nice UI

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td>Orientation Sensor</td>
<td>Displays the orientation of the phone</td>
</tr>
<tr>
<td>Sensor</td>
<td>Brightness Sensor</td>
<td>Displays the brightness</td>
</tr>
<tr>
<td>Sensor</td>
<td>GPS Sensor</td>
<td>Displays the current location of the phone</td>
</tr>
<tr>
<td>Actuator</td>
<td>Light Actuator</td>
<td>Allows turning the camera light of the phone on and off</td>
</tr>
<tr>
<td>Actuator</td>
<td>Picture Actuator</td>
<td>Allows taking pictures with the phone</td>
</tr>
</tbody>
</table>
6.0 Grade Example

- Task 5: Take a picture
  - Uploaded in real time
Server Sockets (TCP)

- Socket definition: Local Addr. + Local Port + Remote Addr. + Remote Port
- Server waits for incoming connections on a specific port
- Server creates **one socket for each client** (“binds a client”)
- These sockets **share the same local socket address**

- Datagram Sockets cannot be “bound” since UDP is connectionless!
Now have fun with the open project!