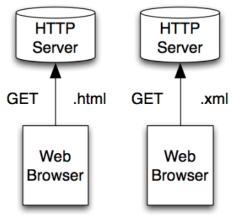
Lecture 9
XML & Web Services

Midterm Review

XML

- XML is great for data exchange
- Our model so far is very browser-centric



- Interactive use, little remote computation

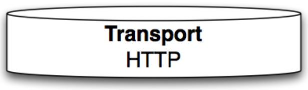
Web Services

- XML can hook up arbitrary software
  - "Web Services"
- Uh, isn't that RPC?
- XML adds:
  - Standard message format
  - Standard data exchange format
  - Some amount of "self-description"
- Web Services also often:
  - Programmatic (non-interactive)
  - Discoverable
  - Described using standard language

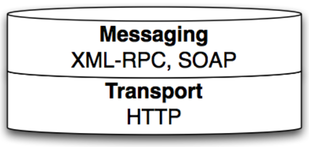
Service Oriented Architecture

- Three parties
  - Service provider
  - Service consumer
  - Service broker
- Provider registers with broker
- Consumer asks broker to find provider

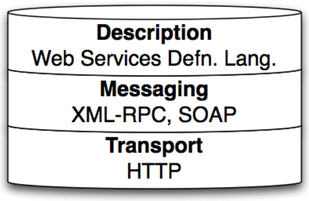
Web Services Stack



Web Services Stack

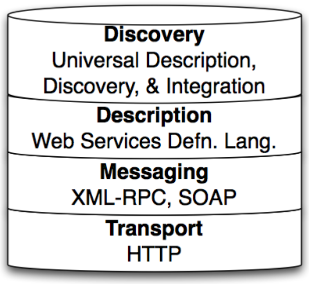


Web Services Stack



7

Web Services Stack



8

Messaging

- Transmitting network requests
  - (aka, "RPC")
- XML-RPC
  - Created in 1998, grew into SOAP
  - Very basic

9

XCoffee

- Want to call
  String order(String cname, int qty)?

  -
    ```
    <?xml version="1.0"?>
    <methodCall>
    <methodName>
    coffee.order
    </methodName>
    <params>
      <param>
        <value><i4>3</i4></value>
        <value>
         <string>Nairobi</string>
        </value>
      </param>
    </params>
     </methodCall>
    ```

10

XCoffee

- Want to hear back from
  String order(String cname, int qty)?

  -
    ```
    <?xml version="1.0"?>
    <methodResponse>
    <params>
      <param>
        <value>
         <string>In stock</string>
        </value>
      </param>
    </params>
     </methodResponse>
    ```

11

XCoffee

- Also possible:

  -
    ```
    <?xml version="1.0"?>
    <methodResponse>
    <fault>
      <value>
        <struct>
        <member>
         <name>faultCode</name>
         <value><i4>22</i4></value>
        </member>
        <member>
         <name>faultString</name>
         <value>
           <string>None left</string>
         </value>
        </member>
        </struct>
      </value>
    </fault>
    </methodResponse>
    ```

12

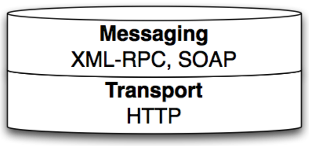
XML-RPC Summary

- You still need application-specific structure; why not just use XML over HTTP?
- Pros
  - XML-RPC auto-generated by many libraries for RPC stub code
  - Anonymous args possible
  - Adds types, which DTDs don't enforce
- Cons:
  - No app-specific DTD to enforce
- Pretty marginal gain

13

SOAP

- Simple Object Access Protocol



- "Sequel" to XML-RPC, SOAP does more & is slightly more complicated

14

SOAP-Mania

- Runs on several protocols, including HTTP, and SNMP OK, too (!!)
- Possibly asynchronous, dep. on protocol
- Three components:
  - Envelope, handles addressing & schemas
  - Header, handles message history
  - Body, handles content
- But first, a detour

15

XML Namespaces

- Optional for XML; mandatory for SOAP
- Allows XML files to refer to uniquely-named elements & attrs
  - `name`, `id` are v. common
  - Parties may want to agree on vocabulary
-
```
<h:table
xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```
- A *namespace* provides scope for labels
- Referred-to by *namespace URL*
- Many standard namespaces

16

Back to SOAP

- Uses Envelope, Encoding namespaces
- SOAP messages **do not have a DTD!**
- Remember: envelope, header, body
-
```
<soap:Envelope
xmlns:soap="http:…"
soap:encodingStyle="http:…">
  <soap:Header>
    …
  </soap:Header>
  <soap:Body>
    …
  </soap:Body>
</soap:Envelope>
```

17

SOAP Header

- Optional; contains info on message
- **`mustUnderstand`** forces processing or error by remote side
-
```
<soap:Header>
  <m:Transaction xmlns:m="http:…"
    soap:mustUnderstand="1">
    234
  </m:Transaction>
</soap:Header>
```
- Allows async replies to requests, multi-msg replies
- Used in conjunction with "Web Services Addressing" to dispatch msg

18

SOAP Body

- The real payload

-
  ```
  <soap:Envelope
  xmlns:soap="http:…"
  soap:encodingStyle="http:…">
    <soap:Body>
      <m:GetPrice xmlns:m="http:…">
       <m:Item>Apples</m:Item>
      </m:GetPrice>
    </soap:Body>
  </soap:Envelope>
  ```
- Name-based parameter passing (not position-based as with XML-RPC)

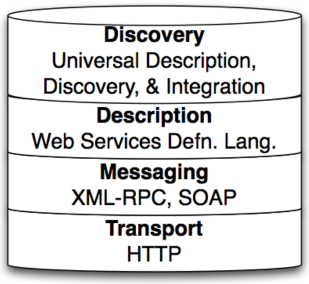
19

SOAP Recap

- You can do a lot more than XML-RPC
  - A lot more complex
  - Arguably, not worth trouble

20

Web Services Stack



21

WSDL

- Web Services Defn/Desc Language
- Used by service-provider to "publish" information on the service
- Provides extremely high-level service description
  - Types: method signatures
    - def'n w/XML Schema
  - Operations: how msgs are combined into service invocations.
    Sender's data elt? Reply's?
    Synchronous or Async?

22

WSDL "operations" Sample

-
  ```
  <portType name="GoogleSearchPort">
    <operation name="doGoogleSearch">
    <input
     message="typens:doGoogleSearch"/>
    <output
  message="typens:doGoogleSearchResponse"
    />
    </operation>
  </portType>
  ```
- portType is equivalent to Java Interface
- Finally, bind portType to real protocols and addresses

23

UDDI

- Universal Description, Discovery, & Integration
- Itself a Web service, UDDI is the WSDL directory
  - Offers "register" and "find" as operations

24

Midterm Topics (1)

- Web Basics
  - URL interpretation
  - HTTP server architecture
  - Client- and server-side dynamic content
  - Model-View-Controller design
  - DOM
- Protocols
  - HTTP
  - TCP - reliability, flow ctrl, congestion ctrl
- Sessions
  - URL encoding
  - Cookies
  - Log-in systems

25

Midterm Topics (2)

- Security & Cryptography
  - Different threat types
  - Message secrecy
  - Authentication
  - Hashing and message integrity
  - Symmetric and assymetric cryptography
  - Public key infrastructure
- XML
  - Data model & parsing
  - DTDs
  - XPath, XSLT, XQuery

26