Announcements

- Lab 3 due back by October 27\textsuperscript{th} (Monday), 11:59pm
  - Will reset the submission form later today

Remainder of the course

- 3 labs (integrated, essentially a “mini-project”, to be done individually)
  - Lab 4: out October 30\textsuperscript{th} (Thursday), due November 12\textsuperscript{th} (Wednesday)
  - Lab 5: out November 13\textsuperscript{th} (Thursday), due November 26\textsuperscript{th}
  - Lab 6: out November 25\textsuperscript{th} (Tuesday), due December 10\textsuperscript{th}
- Final exam: 20\% of overall grade
  - Will emphasize material after mid-term
- No lecture on Thursday, October 23\textsuperscript{rd}

Web Services

[From Lecture 1]

- Main ideas
  - Applications structured as lightweight components, which expose services
    - Example: A Weather component, which offers a GetTemperature service
      - Input parameter: Zip code
      - Output response: An integer that represents the temperature
  - Services discovered, described, and interacted with using standard protocols
    - UDDI, WSDL, SOAP, all of which make heavy use of XML
- Goal: Provide a simple application-to-application interface just like the web has provided a simple human-to-application interface
  - Specifications such as HTML and HTTP, servers and browsers
- What does all this mean?

Sockets, RPC, Distributed Objects … Web Services

Sockets

- Discovery [which, where]
  - Server program must be listening at a well-known port
- Description [what]
  - Out-of-band agreement between application components
- Interaction [how]
  - Low-level networking protocols such as UDP, TCP
  - Unstructured byte streams
  - “Stateful”
Remote Procedure Calls

- **Discovery**
  - Server program identified by “name”, name server maps to location
  - Client must still know about the host

- **Description**
  - Common interface defining the RPC protocol
    - Parameters and return structures
    - Function signatures

- **Interaction**
  - Clients and server stubs facilitate procedure-call like interactions
  - All message passing is hidden from the application writer
  - Remote calls communicate structured data
  - Stateless interaction
    - Remote calls may be handled by one or more server-side instances

Distributed Objects (.NET Remoting)

- **Discovery**
  - Explicit: RPC-like name-to-instance mapping
  - Implicit: Portable object references permit clients to interact with server programs without knowing their location

- **Description**
  - RPC-like common interface, augmented with ...
  - Run-time type inspection of object reference
    - However, comes at the cost of language/CLR dependence

- **Interaction**
  - RPC-like procedure calls
  - Implementation favors local network interaction
  - Rich support for state management
    - Singleton, single call, explicitly marshaled, client-activated objects … leases

Web Services architecture provides XML-based, language-neutral standards for

- **Discovery** [UDDI, WS-Inspection]
  - Approximate location-independent nature of object references in distributed object systems by relying on intermediate brokers, who store/categorize/provide information about services

- **Description** [WSDL]
  - Approximate run-type type inspection by encoding the service types/interface into an XML document that can be interpreted by clients

- **Interaction** [SOAP]
  - RPC-like procedure calls + asynchronous invocations
    - Implementation uses standard, interoperable protocols (HTTP)
  - Goes back to stateless nature of RPC systems
    - Simpler to support, particularly when loosely-coupled services come from multiple owners

Web Services: Importance of Standards

- **SOAP, WSDL, UDDI, WS-Inspection, …**

  - Encode the protocols by which a service created by one user can be used by programs written by another (interoperability)
    - Assembling an individual application becomes easier
    - Everybody benefits, because more services to assemble applications from

  - Same underlying rationale/advantages as in the standardization of human browser-web site interactions
    - HTTP is the protocol (neutral to which entities are implementing it)
    - HTML is the encoding of data
    - Allows seamless access to each other’s web sites
Introducing SOAP, WSDL, and UDDI

• SOAP: Simple Object Access Protocol
  – XML-RPC-like request/response protocol +
  – Support for asynchronous invocations
  – Encoding of additional information in the message
    • Security tokens for authentication/encryption,
    • Message route information, …

• WSDL: Web Services Description Language
  – RPC, Distributed Objects-like common structs/interface +
  – Support for asynchronous invocations
  – Possibility of language-neutral (and automatic) interpretation
    • Web-services tools use WSDL description of a service to automatically
      generate a SOAP-capable proxy

• UDDI: Universal Description, Discovery, and Integration
  – Defines ways of mapping service “characteristics” to service providers
  – “characteristics” generalize “names”

Web Services in .NET using Visual Studio

[ Code walkthrough using Remote Desktop ]

Building and deploying a simple web service
• Setting up the service class and methods
  – [WebMethod] and [WebService] attributes
• Invoking its functionality
• Inspecting its description and discovery interfaces

Writing a web services client
• Adding a “web reference”
• Inspecting automatically generated code for proxy
• Instantiating and using the proxy

Understanding the implementation
• SOAP message exchange
• IIS mapping of service URL