Web Services and Applications

Robert Grimm
New York University
Where I’m Coming From

- General research interests
  - Operating and distributed systems
  - Interaction of programming languages and systems
- PhD, University of Washington, Fall 2002
  - “System support for pervasive applications”
    - A distributed systems architecture
  - Own data model based on tuples
  - Own execution model based on asynchronous events
    - Own networking protocols
  - Hard to interoperate with other distributed systems
    - World Wide Web
Web Services and Applications

- It’s all about the World Wide Web
- But, with a shift in focus
  - Away from human-centric web
    - People access information manually
  - Towards automated web
    - Computers access information automatically
An Example

- Shopping on the Internet
  - Directory web site
    - Find fitting item (say, digital camera)
    - Find best price
  - Vendor web site
    - Buy item (register, log in, select item, check out)
  - Shipping company web site
    - Track package (again and again)

- Automate process
  - Human selects item
  - Shopping agent does the rest
What’s Needed

- Some way to
  - Find services
    - Universal Description, Discovery, and Integration (UDDI)
  - Describe services
    - Web Service Description Language (WSDL)
  - Invoke services
    - XML-based Remote Procedure Calls (XML-RPC)
    - Simple Object Access Protocol (SOAP)
  - Communicate with services
    - HyperText Transport Protocol (HTTP)
Web Service Properties

- Internet accessible
- Standardized messaging
  - XML-based
  - Not tied to single OS or programming language
- Self-describing
- Discoverable
Course Goals

- Three goals
  - Understand web technologies
  - Hatch ideas for research
  - Develop methodology for building complex systems

- Focus on
  - Programming model
    - Also, data model
  - Engineering of large-scale systems
Course Components

- Three components
  - Readings to introduce topics
  - Class discussions to deepen understanding and develop new ideas
  - Programming assignments to provide hands-on experience

- Think
  - Research seminar
    - Learning by reflection and discussion
  - Systems building course
    - Learning by getting your hands dirty
You Are in the Wrong Class If...

- You want to “just” learn SOAP, WSDL, UDDI
- You want to use Microsoft’s .NET or Sun’s ONE
- You don’t have Java programming experience
  - Socket-based, multi-threaded code
- You are not ready for a significant commitment
  - Readings
  - Class
  - Programming assignments
Readings and Class
Readings and Class

- 2-3 required papers per week
  - Write one paragraph summary (per paper)
    - Main idea
    - Innovations (if any)
    - Your criticisms
    - Possible research directions
  - Participate in class discussion
    - I provide slides to review material and guide discussion
- Further readings for additional background
  - If interested, in your copious free time
Topics

- HTTP
  - The evolution of the web protocol
- Building fast servers
  - How to make a single server support a large population
- Clusters
  - How to use many off-the-shelf computers to support an even larger population
- Caching
  - How to avoid going all the way across the Internet
- Content: XML
  - *The* Internet data model
- Content: Multimedia
  - Most of the real data
- Small devices
  - How to network resource-constrained devices
- RPC
  - How to invoke remote services
Topics (cont.)

- **Descriptions**
  - How to describe services

- **Discovery**
  - How to find services

- **Active Everything**
  - How to run code everywhere

- **Representational State Transfer (REST) vs. SOAP**
  - Are SOAP/WSDL/UDDI really the right approach?
What We Don’t Cover

- Content delivery networks (think Akamai)
- Peer-to-peer systems
- Data management systems
- Security
- Economics and Law
  - Micro-payments
  - FatWallet.com sued by Wal*Mart, Target, Best Buy, Staples, OfficeMax, Jo-Ann Stores, KMart
Programming Assignments
Programming Assignments

- Written in Java
- By groups of 4 students
  - 2 students for server
  - 2 students for client
- Using pair programming (only if you want to)
  - Two students design/code/test with one computer
Basic idea: Two programmers share all their work
  - One driver at the keyboard
  - One reviewer actively looking “over the shoulder”
  - Roles are repeatedly switched

Some evidence for better productivity
  - Faster than two solitary programmers
  - More ideas than two solitary programmers
  - Fewer bugs

See Williams & Kessler: All I Really Need to Know about Pair Programming I Learned in Kindergarten
Three More (Required) Tasks

- Test for interoperability
- Track your efforts
- Document your results
This class focuses on standardized protocols

- Testing your group’s client with your group’s server is not enough
- But don’t want to unleash (potentially) buggy code onto the Internet at large

Therefore

- Each group needs to test its client with another group’s server and vice versa
Track Your Efforts

- Time spent on
  - Preparation
  - Design
  - Implementation
  - Basic testing and debugging
  - Interoperability testing
  - Documentation and write-up
- Lines of code
- Number of bugs
  - When introduced and when fixed
Document Your Results

- Ideally, a 3-5 page extended abstract per assignment
- Provide an overview of your goals and design
- Convince me
  - Your client and server work
    - Comply with standards, handle error cases correctly
  - Your client and server meet your design goals
    - Perform a certain way
  - Other group’s client and server work
- Report your project’s statistics
- Share interesting anecdotes (e.g., what was surprising)
The Five Assignments

- HTTP/1.0
- HTTP/1.1
  - Persistent connections
  - Pipelining
  - Digest authentication
- XML Processing
- SOAP
- Your web application
The Five Assignments with Deadlines

- In 3 weeks: HTTP/1.0
- In 5 weeks: HTTP/1.1
  - Persistent connections
  - Pipelining
  - Digest authentication
- In 7 weeks: XML Processing
- In 9 weeks: SOAP
- By end of term: Your web application
Some Application Ideas

- Discussion board
  - Think: Paper summaries
- Time tracker
  - Think: Productivity tracking
- Web cam proxy
  - Think: George Orwell or JenCam
- Visitor announcement and tracking
  - Look at 7th floor lobbies at 715 Broadway
Our Web Cams

- Axis web cams with embedded web server
  - Directly connected to Internet
  - Support only up to 10 simultaneous connections
- One camera overlooking Manhattan
- Two cameras on the 7th floor at 715 Broadway
  - Not connected yet
Your First Assignment: HTTP/1.0

Consult
• HTTP Made Really Easy
• HTTP/1.0 Standard
Follows request/response model
  - Each request from client to server receives a response from server to client
    - Request: Client → Server
    - Response: Client ← Server

Layered on top of TCP
  - One connection per request/response interaction

Performs methods on resources
  - Resources named by URLs
  - Methods are GET, HEAD, and POST
The Three HTTP/1.0 Methods

- **GET**
  - Retrieve the contents of a resource

- **HEAD**
  - Just like GET, but the contents are not returned

- **POST**
  - Add data to a resource
    - Don’t implement without authentication!
Structure of Requests and Responses

- Initial request/response line
  - GET /path/to/file/index.html HTTP/1.0
  - HTTP/1.0 200 OK

- Additional header lines
  - User-Agent: Mozilla

- Optional message body
  - Form data
  - HTML page
The Client

- Opens TCP connection to server (usually port 80)
- Sends request
- Reads response
- Processes response
  - E.g., displays resource contents to user
The Server

- Listens for TCP connections
  - Accepts client connections
- Reads request
- Processes request
  - E.g., reads file
- Sends response
- Closes TCP connection
Composing Clients and Servers: The Proxy Principle

- Interpose on a client/server interaction
  - Without proxy: Client ↔ Server
  - With proxy: Client ↔ (Server+Client) ↔ Server
- Enable
  - Transformation
  - Aggregation
  - Caching
  - Customization
Some Design Trade-Offs

- **Performance and scalability**
  - Event-based servers tend to perform/scale better
    - Java 1.4 supports asynchronous I/O (java.nio)
    - See Flash and SEDA papers from syllabus

- **Resource consumption**
  - Memory, number of threads, CPU consumption under increasing load

- **Extensibility and configurability**
  - Modularity of server
Some Design Trade-Offs (cont.)

- Forgiveness
  - Be liberal in what you accept, be conservative in what you send
    - E.g., always terminate lines with CRLF but accept lines only terminated with LF
  - But, many denial of service attacks exploit this credo
    - ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    - User-Agent: Slow sender
Pick Your Goals

- Design and implement server to meet those goals
- Write your client to
  - Test for protocol conformance
  - Test whether server meets goals
    - Performance
    - Scalability under increasing load
    - Resource consumption under increasing load
    - Performance under attack
With a Little Help from...

- **InternetDate**
  - Parses and formats date/time
- **MimeTypes**
  - Parses and formats data/time
- **InternetInputStream**
  - Reads lines
- **InternetOutputStream**
  - Writes lines
- **InternetHeader**
  - Parses and formats HTTP headers
In Closing...
Collaboration Policy

- Discuss readings and topics with each other
- But write reading summaries individually
- Help each other with programming questions and interoperability testing
- But do not share code outside groups
  - No sharing between groups
  - No code from outside resources (such as Internet)
  - No java.net.URL, java.net.URLConnection
One web site
  - http://www.cs.nyu.edu/rgrimm/teaching/sp03-web/

Two mailing lists
  - g22_3033_008_sp03@cs.nyu.edu
  - g22_3033_008_sp03-readings@cs.nyu.edu
  - Subscribe to both lists
  - Post only plain-text messages, not HTML

x groups
  - Start forming groups now, notify me by Friday