Web Architecture

Servers on the Web

The World-Wide Web

Transfer of resources using HTTP

Web Servers

Server Socket – Listens and generates TCP connections
Opens a socket for each connection

Web Servers

Must deal with many concurrent requests

Content negotiation
Compression
Dealing with large files – chunked transfer
Virtual hosting
Bandwidth throttling

Web Servers

Static content
Session support
Path translation
Interaction with the environment, e.g., OS, DB

Web Servers

Popular Web Servers

- Apache – Apache is the most popular web server (~65% of the web)
- Microsoft – IIS (Internet Information Server) second in popularity (~15% of the web)
- Google – GWS (~5% of the web)
- Their ‘toy’ or complementary brothers
  - Apache Tomcat – will be used in this course
  - Microsoft PWS – a free Personal Web Server for old Microsoft’s operating systems
- Other lightweight servers: nginx, lighttpd
Proxy Server (Web Proxy)

- Server Socket – Listens and generates TCP connections
- Opens a socket for each connection

Does a proxy need access to a DNS?

The Functions of a Web Proxy

- **Cache** – speeds access to resources
- **Anonymity** – keeps the machines behind it anonymous to the server
- **A gateway** – allows changing protocols (e.g., from FTP to HTTP), encodings, etc.
- Allows users to **share a single connection** to the Internet

The Functions of a Web Proxy

- **Filter**
  - Filters outgoing requests (e.g., by using blacklists, keywords, MIME types)
  - Filter ingoing requests (Firewall)
- **Track activities** (intercepting proxy)
- **Access control** – different users receive different access permissions to different parts of the web or to different resources

Applications on The Web

- In the usual (non-REST) approach we can use the web for applying RPCs
- Dynamic pages are the result of applying remote applications
- It is possible to specify the application and the parameters in the GET request or send them in a POST request


Web Services

- When calling a remote web service:
  - The client sends the method name and the parameters inside an envelope – i.e., wrapped in XML, in the body of a POST request
  - Receives the result wrapped in an XML envelope
  - Uses SOAP (will be taught later in the course)
REST

- A design pattern for implementing network systems
- Provides a set of design principles

Resource

- The web is a collection of resources
- A resource has a
  - URI
  - Content
- A resource can be represented in different ways
- A response provides a representation of a resource
- Can we compare two resources by comparing their content?

Different Representations of a Resource

- Consider an HTML page `cs236369.html` and the following files:
  - The compression of `cs236369.html`
  - The results of “fixing” `cs236369.html` to conform to the XHTML standard
  - The presentation of `cs236369.html` using different CSS stylesheets
  - The file `cs236369.html` in different character encodings
  - The file `cs236369.html` in different languages
- Which of the above is the same resource as `cs236369.html`?

Client Interaction

- The client references a web resource using a URL
- A representation of the resource is returned
- The representation places the client in a new state
- When the client selects a hyperlink it accesses another resource
- The new representation places the client application into yet another state
- Thus, the client application transfers state with each resource representation

Representational State Transfer

"Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."

- Roy Fielding

Services

- In REST, there is a resource for every service
- There is a URL for every resource
- So, how do we call a service?
Application Invocation

RESTful
http://university.edu/students/55456

non-RESTful
http://university.edu/getStudent?id=55456

List of Students

RESTful
http://university.edu/students/

<?xml version="1.0"?>
<Students>
  <Student id="55345" href="http://www.university.edu/students/55345"/>
  <Student id="55346" href="http://www.university.edu/students/55346"/>
  <Student id="55347" href="http://www.university.edu/students/55347"/>
  <Student id="55348" href="http://www.university.edu/students/55348"/>
</Students>

To return a list of resources, it provides a list of URIs

Too Many Addresses?

Q: If we have 100,000 students, does this mean we need 100,000 web pages? And what if we will want to represent 100 grades for each student?
A: We just need the method to generate/retrieve the representation of the resource upon request

Operations

All interactions between a client and a web service are done with simple HTTP operations:
- Retrieve information (HTTP GET)
- Create information (HTTP PUT)
- Update information (HTTP POST)
- Delete information (HTTP DELETE)

Requirements

- Architecture in REST is required to provide
  - Separation between clients and servers
  - Stateless protocol
  - The ability to cache responses
  - Mediators between clients and servers (e.g., proxies, gateways) should be transparent to users

Benefits of the RESTful Approach

- What are the benefits of using the REST architecture?
- Easier caching:
  - For example, are the following two URIs represent the same resource:
    - http://university.edu/getGrade?student=111&course=333
    - http://university.edu/getGrade?course=333&student=111
    - http://university.edu/students/111/course/333/grade
Benefits of the RESTful Approach

- Going back and forth is simply moving from one state to another
- What happens in a web shopping application when
  - You put items in the shopping cart
  - Leave the cart untouched (and the browser’s window, as well) for a week
  - Tries to continue the purchase
- Is it possible to add as a bookmark a shopping “state”?

- When a request is sent to an application, on the way to the application, only the headers of HTTP requests are being examined – thus, in REST it is easier to enforce an access control policy or distinguish between different requests
  - For example, if it is forbidden to access some resources that appear in a blacklist, it is easy to enforce that when the resources are specified in the head of the message (the URL in a RESTful approach) and difficult if they are in the body (a SOAP message)

For More Information

- [http://www.ics.uci.edu/~fielding/rest_arch_style.htm](http://www.ics.uci.edu/~fielding/rest_arch_style.htm)