CS640 Lecture 17: 11/04/2014

Announcements
● Programming Assignment 3 Released -- Due Thurs, Nov 13 @ 11:59pm

Using SCP

Web
● Original goal of the web: mechanism to organize and retrieve information
● Inspired by hypertext -- one document links to another
  ○ HyperText Markup Language (HTML) -- used to define basic content and layout of a web page
  ○ Supplemented by Cascading Style Sheets (CSS), JavaScript, images, documents, Flash/Silverlight, and other files
● Uniform resource locator (URL) specifies location of an object
  ○ Perform DNS lookup to obtain IP address of web server to contact
● Client and web server communicate using HTTP

HyperText Transfer Protocol (HTTP)
● Runs atop TCP
● Plain text messages in a request/response sequence -- lines terminated by \n
Request
○ Start line
  ■ Method to execute
    ● GET -- retrieve document
    ● HEAD -- retrieve metadata about document
    ● POST -- send data to server
    ■ URL -- may exclude DN and put this in an option
    ■ HTTP/1.0 or HTTP/1.1
  ○ Options/parameters
    ■ User-Agent -- browser name/version, OS name/version
    ■ Host -- DN portion of URL
    ■ Cookie
  ○ Blank line
  ○ Data -- only for methods like POST

Reply
○ Start line
  ■ HTTP/1.0 or HTTP/1.1
  ■ Status
    ● 200 OK
    ● 403 Forbidden
    ● 404 Not found
    ● 301 Moved permanently
  ○ Options/parameters
    ■ Content-Length
    ■ Content-Type
    ■ Server -- server name/version
    ■ Cache-Control -- how long object can be cached
Example Fetching a Web Page

- DNS lookup
- Establish TCP connection
- Send HTTP request
- Receive HTTP reply
- Close TCP connection
- Parse HTML
- Establish TCP connection
- Send HTTP request for image
- Receive HTTP reply for image
- Close TCP connection
- ...request other objects in page
- ...perform more DNS lookups if objects (e.g. ads) are in different domains (e.g., CDN)
- ...render page while other objects are being fetched

Inefficiencies in HTTP

- Problem: Using a separate TCP connection for each object in a web page has a lot of overhead for connection setup and teardown

For each object: 2 RTTs for connection setup plus at least 1 RTT for fetching data
Solution: HTTP 1.1 introduced persistent connections
  - Exchange multiple request/response messages over the same TCP connection
    - Only need to establish one connection to each server providing content for a page
      - If content is coming from multiple servers (e.g., main page and ads come from 2 different domains), you still need > 1 connection
    - Also benefits throughput
      - For each connection, congestion window starts at 1 and is increasingly exponentially using slow start
      - Takes lots of RTTs to reach maximum throughput -- 8 RTTs for 1 Mbps link; 15 RTTs for 100Mbps link
      - Using one connection means initial slow start only occurs once -- still invoke slow start later if timeout occurs due to loss, but ideally losses are handled through fast retransmit/fast recovery where slow start is not invoked
    - Challenge: how long should a connection stay open?
      - Overhead at server to maintain connections for 1000s of clients
      - Throughput benefits far outweigh this overhead

SPDY
  - Web content transfer protocol designed by Google
  - Further improves upon persistent connection ideas of HTTP 1.1 by allowing multiple requests to be issued at once -- rather than issuing one request, waiting for a response, then issuing the next request, etc.

Web Servers
  - Past: mostly in private data centers and public co-location centers
  - Today: hosted in public clouds
  - Multiple tiers -- front-end, business logic, data store

Web Services
  - Designed for application-to-application communication -- rather than human-to-app communication
    - e.g., Google Maps needs to obtain bus schedule info from Madison Metro
  - Many possible apps that need a protocol for communication
    - Need some technologies to simplify and automate protocol design & implementation
  - Two common architectures: SOAP and REST
  - SOAP (not an acronym)
    - Generate protocols customized to each app
○ Web Service Description Language (WSDL)
  ■ Language for specifying app protocol -- specify format of data and supported operations
  ■ Procedural operations
    ● Message Exchange Pattern (MEP) determines sequence of messages -- e.g.,
      in-only (client to server), or in-out (client request and server reply)
  ■ Define message format using eXtensible Markup Language (XML)
    ● Standard mechanism of encoding messages for transmission --XML, or Fast Infoset (“binary-like” representation)
  ■ Messages can be sent/received using SOAP or HTTP
  ■ Specifications can be reused and combined
○ SOAP
  ■ Used to define transport protocol with exact features an app needs -- e.g.,
    authentication, correlation, reliability, etc.
  ■ Message is an envelope
    ● Header contains blocks for each feature
    ● Body is message, described above
○ Software tools to translate specifications into code
● REST -- REpresentational State Transfer
  ○ Treat individual services as web resources -- identified by URLs and accessed via HTTP
  ○ Runs directly atop HTTP
    ■ Issue GET and POST requests
    ■ HTTP payload is message
  ○ Data-oriented or object-passing, as opposed to procedural operations
  ○ Define document structure using JavaScript Object Notation (JSON) or XML