Networks and Database Systems

Professor Tom Ellman
Lecture 11
Server Pushing Data to Applet

- HTTP is a client/server protocol.
- Each interaction is initiated by the client.
- Sometimes a server needs to continuously send ("push") data to a client:
  - E.g., A web application that monitors stock prices.
  - E.g., A web-based chat program.
- Can be implemented using TCP sockets in an HTTP server.
Two Approaches

- **ServletPushesToAppletWebApp**:  
  - Servlet accepts connections from applets.  
  - Uses TCP, but not HTTP.

- **ServletPushesToAppletWebAppAlt**:  
  - Applets accept connections from servlet.  
  - Uses both TCP and HTTP.
ServletPushesToAppletWebApp

- **Initialization:**
  - Start DataPusher thread. (Pushes data.)
  - Start SocketManager thread. (Accepts connections.)
  - Start Observer thread. (Handles disconnect requests.)

- **Processing of doGet and doPost (nothing).**

- **Destruction:**
  - Stop DataPusher, SocketManager, Observer threads.
  - Close ServerSocket and any open Sockets.
Client Connects to Server Socket in Server

Client

Connect

Server

Make Server Socket

Accept

Push

Push

Bye
ServletPushesToAppletWebAppAlt

• Initialization:
  – Start DataPusher thread. (Pushes data.)

• Processing of doGet and doPost:
  – Make connection command.
    • Connect to client at client-designated port.
    • Store socket and associated print writer in static containers.
  – Break connection command.
    • Close socket connection to client.
    • Delete socket and print writer from static containers.

• Destruction:
  – Stop DataPusher thread.
  – Close open sockets.
Server Connects to Server Socket in Client

Client

HTTP Request (Port Number)

Make Server Socket

Accept

Server

Connect (Port Number)

Push

Push

Bye

Bye
Comparing Approaches

• Client Connects:
  – Simple client / Complex server.
  – Server must create ServerSocket and thread to monitor it, even if no clients ever use it.

• Server Connects:
  – Simple server / Complex client.
  – Client creates ServerSocket it will use immediately.
JavaScript and Ajax

• JavaScript:
  – Script may be placed in web page.
  – Execution of script makes web page dynamic.
  – No threads or sockets in JavaScript.
  – JavaScript may initiate HTTP requests.

• Ajax:
  – Framework for building dynamic web pages.
  – Issue HTTP request for updated page data.
  – Use DOM operations to modify only parts of the web page.
JavaScript / Ajax Example

• JavaScript code is embedded in web page.
• On script execution:
  – Create XMLHttpRequest and set target URL and headers.
  – Register call back function on “ready state change” event.
  – Send request.
  – Exit script.
• When request is filled:
  – Call back function is invoked, passing request result.
  – Call back function modifies component of web page.
<html><head><title>AJAX Example</title> </head><body><center /> <h1>Loading a web page into a DIV</h1> <div id='info'>This sentence will be replaced</div>

<script>
params = "url=amazon.com/gp/aw"
request = new ajaxRequest()
request.open("POST", "urlpost.php", true)
request.setRequestHeader("Content-type","application/x-www-form-urlencoded")
request.setRequestHeader("Content-length", params.length)
request.setRequestHeader("Connection", "close")
request.onreadystatechange = function() {
  if (this.readyState == 4 &&
      this.status == 200 &&
      this.responseText != null) {
    document.getElementById('info').innerHTML = this.responseText
  }
}

request.send(params)

// ... Omitted

</script></body></html>
Server Pushes to JavaScript/Ajax Page?

- Strictly speaking, it’s not possible.
- Since all HTTP request must be initiated by the client.
- Comet is a framework for faking it.
  - Server sends HTTP response gradually, in pieces.
  - Client reads each piece and updates page, before getting the entire response.
Client initiates the request, indicating a long time limit on how long it will wait for the complete response. Server sends the response in pieces, spread out over hours, days, etc. Clients reads each piece and immediately updates the web page.
Virtual Peering

• Problem: Applets can normally open socket connections only with the host from which the applet was served.
• What if we want to open a socket between two applets running in browsers on separate hosts?
• Peer proxy server acts as intermediary.
Design Challenge

• Design the applet (client) and servlet (server).
• What commands are sent via HTTP?
• What sockets are opened? In what roles?
• What threads do client and server launch?
• What data structures do they maintain?
• How does server function as intermediary?
Private Channel in Chat Room?

A browser will not allow an applet to communicate with any host except the server from which the enclosing web page was loaded (unless the applet is digitally signed and the user agrees).
Server as Proxy for Private Channel

Client1

Accept (Port)
Accepting (Client2, Port)

Server

Accept (Port’)
Client1 Accepting (Port’)

Connect (Port)

Message12

Message21

Message12

Client2

Connect (Port’)

Message21
Server as Proxy for Private Channel

- Set up two sockets:
  - One socket between client1 and server. (S1)
  - One socket between server and client2. (S2)
- Server virtually connects the two sockets two each other:
  - Incoming message on S1 is sent through S2.
  - Incoming message on S2 is sent through S1.
Problem

• Server uses two sockets for each pair of privately communicating clients.
• Total of M*N sockets for N clients, each communicating privately with M friends.
• Solution:
  – Server and client use one socket for communicating with all the client’s friends.
  – Source and destination must be encoded in data passed through the socket.