

Problem Set 2: HTTP, AJAX, DNS, and XMPP

Due date: **Wed, Mar 18, 11:59pm**

Submission: In PDF format via the submission form on the website. The problem set accounts for 100 points.

1. HTTP Protocol (20 points)

Do the HTTP Wireshark Lab v2.0 available at

http://courses.cs.vt.edu/~cs5565/spring2009/local/wiresharklabs/Wireshark_HTTP_July_22_2007.pdf. This lab asks you to use Wireshark to look at the traffic produced by a web browser. You answer a series of simple questions about it and submit the answers. This lab should give you a good understanding of how HTTP works.

2. Server-centric AJAX (30 points)

Use either Firebug (<http://getfirebug.com/>) or Wireshark to reverse-engineer the server-centric AJAX framework ZK. You may find sample ZK applications here: <http://www.zkoss.org/demo/> and here: <http://libx.org/editionbuilder>.

Prepare a report of 2 pages or less that concisely describes the basics of the update protocol used by ZK's update engine. Include a specific example of an interaction and discuss how the protocol encodes necessary updates to elements on a page.

3. DNS Protocol (20 points)

Do the DNS Wireshark Lab v2.0 available at

http://courses.cs.vt.edu/~cs5565/spring2009/local/wiresharklabs/Wireshark_DNS_July_22_2007.pdf. This lab asks you to use Wireshark to look at the DNS traffic produced when web browsing and when using the tool nslookup. If nslookup is not available, use 'dig' instead. You answer a series of simple questions about it and submit the answers. This lab should give you a good understanding of how DNS works. Feel free to explore scenarios not asked for in the lab.

4. Jabber/XMPP Protocol (30 points)

The Extensible Messaging and Presence Protocol (XMPP), developed by the Jabber community (<http://www.xmpp.org/>), is increasingly being adopted as the standard protocol for instant messaging and other XML-stream based Internet applications. In this lab, you're asked to try it out and get a basic idea of how aspects of it work.

The RFCs that describe XMPP are available at <http://xmpp.org/extensions/>. You should refer to those RFCs for this problem, specifically RFCs 3920 and RFC 3921. Read as much as necessary in those RFCs.

You will need to have a Google Account and the Google Talk client installed for this lab (see <http://www.google.com/talk/>).

1. Start packet capture.
2. Start google talk, and log into your account.
3. Add cs5565bot@gmail.com to your contact list. It should accept this invitation. If it does not, send email to cs5565-staff@gmail.com to check on the status of the bot.
4. Send one or more messages to cs5565bot. It should echo these messages back to you. (Privacy notice: everything you type will be logged.)
5. Log off.
6. Stop packet capture.

In Wireshark, set your filter to "jabber". You should see a number of Jabber Request & Response messages. Take a look at them and answer the following questions:

1. In the opening <stream> element sent to Google Talk, what version of XMPP is being used? What is the value of the "to" attribute?
2. Does Google Talk require the use of TLS? See RFC 3920, 5.2.3
3. How many SASL authentication mechanisms does Google Talk offer and what are they? See RFC 3920, 6.1.3.
4. Is Google Talk in compliance with RFC 3920, 14.7?
5. How does Google Talk signal successful SASL authentication? Copy the XML element used.
6. What resource identifier did Google Talk assign for this session? Look for the <jid> element in the reply to the <iq type="set" id="0"> request sent.
7. How many and which types of <presence> stanzas does the Google Talk client send? (See RFC 3921, 5.5 and 6.1)
8. Copy one <message> stanza your Talk client sent to the server here.
9. Copy one <message> stanza your Talk client received here.
10. Provide a screenshot.

5. Extra Credit (20 pts)

Write your own talk bot. It should contain at least as much functionality as the CS5565 Talk Bot. Use an XMPP client library such as Smack, see: <http://www.igniterealtime.org/downloads/index.jsp#smack>

Arrange with the TA to demo the bot and include your source code in your submission.