Model Answers, Midterm Exam #1
EECS485, Winter 2010

----
1)

In these answers, there is some ambiguity about whether the events take place at the same connection, or different connections within the same browser. Thus, different answers may be acceptable, depending on the reasoning given.

-- (A) if the "text" is considered to be the error message that reflects the HTTP message.
    (B) if the HTTP error comes from a non-textual source (such as images).

-- If on the same connection, then B comes first. If on different connections, there is no expected order.

-- A, as the PHP is most likely executed first to generate HTML, which then allows the browser to ask for the JPEG.
   If the reasoning states that PHP is "initialized" when the server comes up, then (A) is acceptable.

-- A. Not much ambiguity here - there is almost no interpretation by which the SQL db can do anything prior to the connection being established.

2a)

Some possibilities:
-- Performance penalty during each HTTP request
-- Not transportable to other browsers
-- Lose history when cookie cache is reset.
-- Requires encryption to preserve user privacy.
-- Other novel ones may be acceptable.

2b)

-- URL Encoding is the only easy way to get information from one server to another. Cookies and server state are constrained to individual sites.

-- Because the counter is for all the users on a site, it should be stored at the server.

-- Server is the most reasonable. The entire address book should be durable and might be large, making cookies or URL-encoding inappropriate.

-- Cookies are probably the best way, though URL may be acceptable with some reasoning.

3.

A few things we're looking for here:
-- There should be a redirect to the login server.
-- The original target URL should be url-encoded and sent along to the login server.
-- When complete, the login server redirects to the url-encoded value.

Material on what is stored in cookies (login credentials) is very good, but not critical for full credit.

4.
Model should consist of the backend database, plus data model of restaurants and reviews. It executes at the db tier and possibly at the web server tier. It keeps the data safe and coherent.
View should consist of the in-browser display of text, plus HTML-generating code at web server. Executes at browser and web server. Reflects state of the content database.
Controller consists of the text box for adding a review, plus the web app code for processing updates. Executes at browser and web server. Allows user to "poke and prod" the database to the extent the application allows.

One point for each component. Half credit for incomplete but correct answers (or for technically-incorrect but well-reasoned ones).

5.
a) TCP performance will drop due to packet loss. It is admirable, but not necessary, to mention that TCP congestion control could really be sent for a loop by wireless interference that is interpreted as congestion on the network.

b) There are at least two acceptable answers here. One is to use the meta radio as an indicator that TCP should time-out its packets more quickly, resulting in less waiting time after a packet is destroyed due to interference. Impact is likely to be positive for both, perhaps slightly more so for long-lasting transfers, simply because less time will be wasted waiting.

Another is that the meta radio could be used to prevent TCP from interpreting the wireless-interference as in-network congestion (and thus reducing the TCP send rate). Impact is likely to be much bigger on long-lasting transfers, which would otherwise be bogged down at very low transfer rates, due to incorrectly-applied congestion control.

Some people mentioned that the meta radio could be used to turn off the transmitter, or to increase the wait-time. These answers often had some solid reasoning, and so partial credit was appropriate in some cases. However, neither of these actions will increase the device's TCP performance.

c) If the student indicates the TCP time-out issue, then a faulty meta radio results in unnecessary packet timeouts. For the congestion-control issue, a faulty radio results in inaccurate detection of network congestion.

6.

DOM-based is good for XML processing that requires the entire tree, e.g., you want to test whether an element is the uncle of another one. DOM is also a memory hog. SAX allows per-elt processing, but is memory-efficient. The latter is good for message-exchange, the former often good for structure-interpretation (e.g., rendering an XHTML Web page).

If student has a good answer that mentions application scenarios, but not the memory impact, half credit.

7.
Correct order:

B -- A -- D -- C

B has to come first, otherwise we don't know if we can trust the public key needed for the assymetric part of the conversation.
A is appropriate after B is performed.
D is the main goal of using public-key crypto.
C is possible now that the symmetric key has been negotiated.

8a. (2 points) A few variants are possible. One is Priv_C(Pub_C(m), date). Close is Pub_N(Priv_C(m, date)), but in this case the message is not secret from Nancy (partial credit). Other variants might be OK, but Charlie must apply his own signature (via Priv_C()) in such a way that the message is signed and the date is tamperproof.

8b. (2 points) Nancy verifies two things. She verifies Charlie's signature by applying his public key. She also tests the date to see if it is accurate.

8c. (1 point) Nancy outputs Priv_N(Pub_C(m), date). She may or may not strip Charlie's signature from the output.

8d. (1 point) The court can remove the signatures, because that information is public. It can then take any "test-message" given by Charlie, apply Charlie's public key to it, and see if it is equivalent to Pub_C(m). Based on different assumptions about the court's legal power (can it take private keys?), different variants might be acceptable. Note that while the date is necessary for the court to verify the message, students who did not include the date in 8A should not be penalized additionally here.

9.
9a. Does not directly stymie the attacker, because attacker can simply MD5-encode the contents of the dictionary. However, it does prevent the attacker from observing plaintext passwords on the wire.

9b. This has much the same effect as above. Attackers can still connect using a secure connection and attempt to login with dictionary passwords. But eavesdropping is now impossible. The security layer adds a modest computational overhead that will probably not be a serious deterrence.

9c. A 5-second delay on all login attempts is a successful way to deter dictionary attacks, because it limits the effective number of draws from the dictionary that the attacker can use. The attacker can overcome this (at some cost) by issuing many requests simultaneously. A delay on login failures *only* can be somewhat overcome by the attacker by simply disconnecting if the attack is not immediately successful; this latter point is pretty subtle and is not required for full credit.

9d. This is extremely successful in deterring dictionary attacks. System-assigned passwords can be generated to be extremely unlikely to be found in a dictionary. The only possible downside is that the resulting passwords are often hard for users to remember.

----

10. (0.5 points each)

a.
- false
- false
- false
- true
- true

b.
- any author in the xml
- all comment authors who have an email address.
- firstname of all blog authors

c.
- returns the names of all blogs written after 2004. Each element is between tags <blog> </blog>
- returns the body of all posts written after 2007. all elements together are between two tags <post> </post>

11a. (1 points) the validity can only be judged based on an XML schema (a DTD).

11b.

- (1 points)

```
for $d in distinct-values(doc("doc.xml")//actor)
return <name>{$d}</name>
```

```
<name>Oleg Yankovskiy</name>
<name>Kate Winslet</name>
<name>Billy Zane</name>
<name>Leonardo DiCaprio</name>
<name>Russell Crowe</name>
<name>Joaquin Phoenix</name>
<name>Connie Nielsen</name>
<name>Golshifteh Farahani</name>
<name>Ralph Fiennes</name>
<name/>
```

****
- (1 points)

```
for $d in distinct-values(doc("doc.xml")//director)
   let $m := doc("doc.xml")//movie[director=$d] where count($m)>1
return <director>{$d}</director>
```

```
<director>Ridley Scott</director>
<director>Stephen Daldry</director>
```

****
- (2 points)

```
for $m in doc("doc.xml")//movie where count($m//actor[@oscarWinner="yes"]) >=
count($m//actor[@oscarWinner="no"])
   return $m/title
```

```
<title>Gladiator</title>
<title>The Reader</title>
```