EECS 122: Introduction to Communication Networks

Homework 2 Solutions

Solution 1.

a) From the SMTP spec (RFC 821), we see that some of the Received fields are added by the intermediate MTAs, but the rest of the message is created by the originating host, and is therefore likely to be bogus in a spam message. The Received headers form a linked list, because each one names the host that received the message and also the host it came from. If the originating host refuses to put its own name in the header (as we are told to assume), then there will be a break in the chain. Tracing the linked list backwards, starting from the final (and trusted) machine mnemosyne.cs.berkeley.edu, we see that the break happens where mail.everfaster.com says it got the message from gate.hypermoon.com, but there is no Received field containing "by gate.hypermoon.com". Therefore gate.hypermoon.com is a liar, and we were told to assume that only the originating host lies. (Actually, if mail.everfaster.com is not careful about verifying the accuracy of the information it puts in the header, pool37.qs4w.longlink.net and 217.6.1.7 are more reliable identifiers for the originating host.) (The remaining Received fields are bogus, as are the From field and initial From line, so hotdogcity.com, hotmail.com, and aol.com all had nothing to do with the relaying of this message—those names were put there to confuse people [yes, software for sending spam really plays tricks like that]. Ironically, gate.hypermoon.com would have been more convincing in its attempt to place the blame on hotdogcity.com if it had used its real name instead of server.big-hello.com, because then we would not have found a break in the linked list.)

b) Notice that mail.everfaster.com received the message from gate.hypermoon.com, then sent it to cs2.cs.berkeley.edu. Therefore mail.everfaster.com is configured to relay messages toward their destinations no matter where the messages come from. (This was once a common configuration, but in recent years spammers have started taking advantage of such MTAs. It takes a substantial amount of computing and network resources to send thousands of copies of a message from a single machine. But gate.hypermoon.com can send a single copy to mail.everfaster.com with hundreds of addresses listed in SMTP `RCPT` commands, thus using someone else's resources rather than its own. Because of this practice, most MTAs are now configured to relay messages only if they are coming from or going to the MTA's domain.)

c) Complaints should be sent to postmaster@everfaster.com regarding the overly permissive configuration, and to abuse@longlink.net regarding the antisocial behavior of its customer hypermoon.com. Although RFC 2142 documents the conventional

"postmaster" and "abuse" addresses, there was some additional knowledge required to determine the domain longlink.net (which is why this part was worth 0 points). The syntax of the Received field indicates that the name gate.hypermoon.com maps to the IP address 217.6.1.7, which maps back to the name pool37.qs4w.longlink.net. This is a typical situation for very small customers of internet service providers (ISPs)—both the ISP and the customer have their own name for the same address. Sending a complaint to hypermoon.com would probably not be effective, because those are the people who sent the spam, but sending a complaint to their ISP is more likely to have an effect.

Solution 3. According to RFC 2396, the format of an HTTP URL (without the optional query) is

http://*host*:*port*/*path*/*path*/*path*...

In each component of the URL, certain characters are reserved, and must therefore be escaped if they appear. In this example, the spaces, slash (/), and question mark (?) in the last path component must be escaped. Their ASCII codes are 32, 47, and 63 respectively, which are 20, 2F, and 3F in hexadecimal. The result is

http://foo.bar.org:12345/networking/What%20is%20TCP%2FIP%3F

Solution 4.

```
<p>I am doing
<a href="http://www.cs.berkeley.edu/~amc/eecs122/">EECS
122</a> homework.</p>
```

Solution 5.

a) If the message body is being generated by a running program, the HTTP server (and even the program) might not know ahead of time how long the body will be.

b) HTTP 1.1 provides the Transfer-Encoding header field and the `chunked` transfer coding, allowing the body to be broken up into small chunks. (The size of each chunk must be declared before the chunk is sent, but the number of chunks need not be declared at the beginning—the sequence is terminated by a zero-length chunk.)

Solution 7. The FTP `PORT` command is usually used to transfer files between the FTP server and client, or between two remote FTP servers, but can also be used to transfer a file from an FTP server to a different sort of server. In this case, create a file containing the SMTP commands necessary to send a message to an MTA. Transfer this file to the public FTP server (using FTP normally), then use the FTP `PORT` command to instruct

the FTP server to make a data connection to port 25 of a mail exchanger (in other words, the FTP server connects to a mail transfer agent), and finally instruct the FTP server to transfer the file. The SMTP commands convey a message to the MTA, and the MTA records (accurately) that the message was received from the FTP server, not from your machine. (If you try this, it probably won't work, because most FTP servers have been altered to disallow this sort of thing.)

Solution 8.

a) The key is to think of the users collectively, and the news servers collectively. The u users together create articles at a total rate of up. We are told that every server receives every article, and that the servers avoid sending duplicates, which means every server receives every article exactly once. Therefore, the n servers together receive articles at a total rate of nup. Since up of this incoming traffic comes from users, the other $(n-1)up$ must come from servers, which means the servers collectively send at a total rate of $(n-1)up$ to each other. The users collectively receive articles at a total rate of ur, which means the servers collectively send at rate ur to the users. The total traffic sent by the servers (to users and to each other) is $(n-1)up+ur = nup+(r-p)u$. Since $r > p$, the servers send more than they receive, so they will hit their sending limit of m per server before their receiving limit of m per server. The total sending limit for all the servers is nm, so when they reach their limit:

$$\begin{aligned}(n-1)up+ur &= nm \\ u &= \frac{nm}{(n-1)p+r}\end{aligned}$$

b) Substituting $n = 1$ into the result of part (a), we get $u = m/r$ (in other words, as many users as one server can feed articles to).

c) Taking the limit as n increases, we get $u = m/p$ (in other words, as many users as it takes to create articles faster than servers can send them to their neighbors).

d) Substituting $u = m/2p$ into the result of part (a), we get $n = \frac{r}{p} - 1$. (So if users on average read 1000 times as many articles as they write, then adding more servers after the 999^{th} will not go much further in supporting more users. To support more users, you need to make the servers communicate faster, as indicated in part (c).)