Network Systems (201300179), Test 1

February 14, 2014, 15:45–17:15

- Only 1 double-sided sheet of notes / summary (any font size/density!), and a dictionary are allowed as reference material. Use of the book by Peterson & Davie or any other written material is not allowed. Use of a simple (non-graphical) calculator is allowed. Use of laptops, tablets, graphical calculators, mobile phones, etc., is not allowed. ***Please remove any such material and equipment from your desk, now!***
- Although the questions are stated in English, you may answer in English or Dutch, whichever you are more comfortable with.
- You should always explain or motivate your answers, with so much detail that the grader can judge whether you understand the material; so just saying "yes" or giving a formula without explanation is not enough.
- Visiting the toilet without explicit permission of the supervisor is not allowed. During the last 30 minutes of the exam, no toilet visits are allowed.

1. Performance and reliable data transfer

The Curiosity rover is exploring the planet Mars for evidence the planet could have once supported life. When the earth is visible for the rover, it has a radio communication link to the earth of 10 kbit/s (10^4 bit/s). The distance between Mars and earth varies, but for simplicity, we assume it to be constant at 300 Gm ($3 \cdot 10^{11}$ m). Radio communications takes place at the speed of light ($3 \cdot 10^8$ m/s). Recently, the Curiosity had to cross a dune, which is a risky operation.

4 pt (a) After Curiosity has successfully crossed the dune, how long does it take at least for ground control to know this? You may assume that the success or failure of the crossing can be coded in a single bit, and that the communication is error free. Which delay component of the communication is determining this delay?

propagation delay dominates; 1000 seconds

4 pt (b) After crossing the dune, the Curiosity has a terrific view on a yet unexplored terrain. It makes a high definition picture (100 Mbyte, assume for simplicity $8 \cdot 10^8$ bits) of it and sends that to the earth. How long does it take for this picture to be completely received on earth? You may again assume error free communication. Which delay component is dominant here?

dtrans=80000 s dominates; total 81000 s

4 pt (c) Since errors in fact may occur, the picture is transmitted in packets of 1000 bits each, so they can be retranmitted if needed, using a sliding window mechanism. How large should the send window size at least be to do this for optimal efficiency?

RTT=2000s; 1 packet takes 0.1 s, so SWS=20000 sequence numbers is needed to be able to keep the pipe full

2 pt (d) What receive window size would you recommend in this case? Why?

RWS=SWS; otherwise, lost packets could cause later packets to be retransmitted even if they were received correctly at first, which especially with such a large delay bandwidth product would be inefficient

4 pt (e) While several rovers are exploring the surface of Mars, other spacecraft are orbiting Mars. An alternative way for the rovers to communicate with the earth is to communicate via an orbiter. Such an orbiter has radio links with several Mars rovers, and a long-distance communication link to the earth. As such, it functions as a switch, multiplexing data from multiple orbiters onto the link to earth. Suppose you had to design this switch. Would you design it as a packet switch or as a circuit switch? Give advantages and/or disadvantages of both modes of operation, and describe the consequences for the type of multiplexing on the link from the orbiter to earth.

Many aspects can be mentioned here, see also the study material. The main disadvantage of circuit switching to be mentioned here, is the lack of statistical multiplexing: when one rover is not transmitting anything, the link from the orbiter to the earth cannot be fully used.

---

2. Information theory

We're not yet done with Curiosity. Let's assume that while driving on Mars' surface, it analyzes the soil under its wheels once per second. Each such observation has three possible outcomes: *Dry* with estimated probability of 95%; *Water* with 4.999%, and *Life* with 0.001%.

4 pt (a) How much information is there in each observation?

0.2866 bits

While driving, Curiosity may not be able to keep its antenna aimed at the earth. It then resorts to a much weaker radio link, over which only 10 bits per second can be sent, and each such bit has a 40% probability of not being received correctly.

4 pt (b) Suppose we would want to use this link to send one such a soil analysis report per second to earth (and nothing else). How low could the error probability of these reports be made, by using good source and channel coding?

The capacity of this channel is $(1 + .4 * \log_2(.4)/\log_2(2) + .6 * \log_2(.6)/\log_2(2)) * 10 = 0.29$ bits/s. This is more than needed for the reports (as calculated in the previous question), so the error probability can be made as close to 0 as desired.

Let's go back to the messages themselves.

4 pt (c) Propose a way to encode the soil analysis reports in less than 1.1 bits per observation on average, and show that your code indeed achieves this.

0 for Dry, 10 for Water, 11 for Life; takes $1 \cdot .95 + 2 \cdot .05 = 1.05$ bits on average.

4 pt (d) Try to improve on your answer in (c): propose a way to encode the information in less than 0.5 bits on average.

This question requires some ingenuity. The only way to achieve this, is to not try to make codes for single messages, but make codes for groups of e.g. four messages. One possibility is: let a single '1' mean four times Dry; and let '0' mean that what follows is four messages coded according to the previous question.

Proof (but note that the question didn't ask for this): with probability $0.95^4 = 0.8145$, the three messages are all Dry, and thus together take 1 bit. In the remaining cases, at least one message is not Dry and thus takes 2 bits; the other three messages on average again take 1.05 bits each, making the total 1 (for the '0' bit) + 2 (for the non-Dry message) + $3 \cdot 1.05$ = 6.15 bits. So the total average number of bits is $1 \cdot 0.8145 + 6.15 \cdot 0.1855 = 1.955$ bits, for *four* messages; that's less than 0.5 bits per message.

Many students answered something like "send nothing for Dry, '0' for Water, '1' for Life". This indeed reduces the average message length to about 0.05 bits, but it does not work; suppose the receiver receives "00100", how does it know how many times Dry was sent between e.g. the first two Water messages?

---

3. Peer-to-peer applications

2 pt (a) Give in one word the key advantage of peer-to-peer applications compared to classical client-server applications.

Scalability.

In the remainder of this exercise, we consider a typical peer-to-peer downloading scenario. A file of size F (bytes) is distributed from one node (server) to N peers. The upload data rate of the server where the file initially resides is u_s (bytes per second). The download data rate of each peer is d_i (bytes per second); the upload data rate of each peer (apart from the initial one) is u_i bytes per second ($i = 1, ..., N$). Some more notation: $d_{min} = \min_i\{d_i\}$ and $\sum_i u_i = u_1 + \cdots + u_N$.

It is know that the distribution time D in the scenario above can be given by

$$D \geq \max\left\{\frac{F}{u_s}\,,\ \frac{F}{d_{min}}\,,\ \frac{NF}{u_s + \sum_i u_i}\right\}.$$

5 pt (b) Explain the expression above, including the $\geq$ sign and the three terms.

$\frac{F}{u_s}$: D can not be smaller than the time for the server to upload the file into the network (file size divided by server uplink data rate).

$\frac{F}{d_{min}}$: D can not be smaller than the time for the slowest client to download the file from the network (file size divided by the slowest downlink data rate).

$\frac{NF}{u_s+\sum_i u_i}$: D cannot be smaller than the time needed to jointly upload N copies of the file into the network (N times file size divided by the sum of all uplink data rates)

---

4. HTTP

The extremely basic personal homepage of one of the lecturers of this course has the following HTML code. Together with the embedded images, the page is stored on a host running a web server as application.

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Geerts Webpagina</title>
</head>
<body>
<script language="JavaScript" type="text/javascript">
<!-- hide addresses
function compose(name,domain) {
  var link='';
  var address= name + '@' + domain;
  link = 'mai' + 'lto:' + address;
  return link;
}
-->
</script>
<h1>Geert Heijenk</h1>
<img src="gehe_pas.jpg">
<img src="no-facebook-me.png" alt="you won't find me on Facebook" />
<br>
Welkom op mijn webpagina. Als je me een berichtje wilt sturen kan dat naar <a
    href="javascript.html" onmouseover="this.href=compose('geert','heijenk.com')
    "> geert [at] heijenk.com</a>. Op de Universiteit Twente heb ik een webpagina
     met veel meer informatie: <a href="http://www.cs.utwente.nl/~heijenk" target
    ="_parent">www.cs.utwente.nl/~heijenk</a>.<br>
</body>
</html>
```

2 pt (a) This HTML code contains some executable Javascript code. Where will this be executed: on the server, or on the client, or on both?

On the client.

3 pt (b) If HTTP 1.0 (without persistent connections and without parallel connections) is used to fetch this homepage, how many Round Trip Times (RTT) are needed before the complete page can be displayed in the browser? Explain your answer.

1 to open connection, 1 to fetch html page, 1 to open connection for 1st image, 1 to fetch first image, 1 to open connection for 2nd image, 1 to fetch second image. Total: 6 RTTs.

2 pt (c) Answer the same question for the case where HTTP 1.1 (with persistent connections and with pipelining) is used.

1 to open connection, 1 to fetch html page, 1 to fetch 2 images in parallel, total: 3 RTTs.

3 pt (d) In the context of HTTP 1.0, parallel connections are used to improve the performance. Would this also be of use in case of HTTP 1.1?

When using pipelining, multiple HTTP requests can already be sent in parallel (before receiving the response). So, using parallel connections does not improve performance. The only case where using parallel connections could improve performance is if the size of the (receive) window is limiting the performance. In this case, parallel connections may allow the server to have more data oustanding (unacknowledged).