CSEE 4911: Computer Networks Due on Thursday, February 16th, 2012

Assignment #2 – Application Layer

Laurent Charignon, Avner May, Mathias Lécuyer (TAs) A. Chaintreau (instructor),

How to read this assignment : Exercise levels are indicated as follows

($\rightarrowtail$) "elementary": the answer is not strictly speaking obvious, but it fits in a single sentence, and it is an immediate application of results covered in the lectures.

Use them as a checkpoint: it is strongly advised to go back to your notes if the answer to one of these questions does not come to you in a few minutes.

($\curvearrowright$) "intermediary": The answer to this question is not an immediate translation of results covered in class, it can be deduced from them with a reasonable effort.

Use them as a practice: how far are you from the answer? Do you still feel uncomfortable with some of the notions? which part could you complete quickly?

($\nrightarrow$) "tortuous": this question either requires an advanced notion, a proof that is long or inventive, or it is still open.

Use them as an inspiration: can you answer any of them? does it bring you to another problem that you can answer or study further? It is recommended to work on this question only AFTER you are done with the rest!

Exercise 1: DNS queries (6 pt)

In the DNS architecture, suppose that queries to a local DNS server for a newly queried hostname are forwarded first to a Root DNS server, then to a TLD DNS server, and finally to an authoritative server. Suppose further that the root and TLD servers *do not cache* any results whatsoever, other than what they are always supposed to know.

Assume further that the local DNS server, `dns.columbia.edu`, caches all queries that it makes (including to root, TLD and authoritative servers).

Suppose the following sequence of hostnames are sent to `dns.columbia.edu`:

- www.cs.umass.edu
- gaia.cs.umass.edu
- www.mit.edu
- www.cs.umass.edu

1. ($\rightarrowtail$) Taking into account the information that is cached at the local DNS server, describe the sequence of DNS servers contacted for each query when recursive DNS queries are used.

2. ($\rightarrowtail$) What if iterative DNS queries are used?

Exercise 2: Let's see how big the Internet is! (9 pt, including 1pt for the last question)

Motivation: Never underestimate the power of a small random experiment.

Randomly pick 20 Internet addresses from the range 1.0.0.0 to 218.255.255.255. For each address you will check the following:

- whether this address has a DNS name assigned to it (using nslookup or host),
- whether it responds to a ping request (using ping).

Examples: Command `host` *and* `ping`

```
$ host 158.48.3.7
158.48.3.7 does not exist (Authoritative answer)

$ /usr/sbin/ping 158.48.3.7
ICMP Time exceeded in transit from 158.48.1.1

$ host 195.37.76.19
Name: saturn.fokus.gmd.de
Address: 195.37.76.19

$ /usr/sbin/ping 195.37.76.19
195.37.76.19 is alive
```

1. (↷) Provide a table with IP-addresses and your results, together with numerical statistics on the fraction of addresses with a DNS name and the fraction of addresses that are ping'able. If you obtain trivial results (none of them are), we recommend you double the sample.

   NB: We recommend you write a script to automate this process and, feel free to test more addresses!

2. (↷) Using result from the question above, estimate the size of the Internet.

3. (↷) Describe the limitations of this particular approach.

Another approach : Let's turn the problem on its head, pick 20 random words in a dictionary (we recommend using a paper copy, or going to a library), and check whether it is a registered DNS domain in the `.com`, `.edu`, `.org` top-level domain.

4. (↷) Again, provide detailed results in a table together with numerical statistics on the fraction of words that are associated with a domain name.

5. (↬) Using results from above, estimate the size of the Internet. Compare your results to the previous estimation? Are they identical? Should they be? Discuss the limitations of this approach.

   NB: We will collect all statistics and summarize the results of the class.

Exercise 3: Quizz (4 pt)

True or false? Justify your answers!

1. ($\curvearrowright$) Suppose a user requests a web page that consists of some text and two images. For this page the client will send one request message and receive three response messages.

2. ($\curvearrowright$) Two distinct web pages, for example, `http://www.cs.columbia.edu/IRT/`, and `http://www.cs.columbia.edu/home/` can be sent over the same persistent connection.

3. ($\curvearrowright$) The `Date:` header in the HTTP response message indicates when the object in the response was last modified.

4. ($\curvearrowright$) With non-persistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.

Exercise 4: Applications Requirements (3 pt)

1. ($\curvearrowright$) Referring to Figure 2.4 in text, we see that none of the applications listed in the figure require both "no data loss" and "timing". Can you conceive of an application that requires no data loss and that is also highly time-sensitive?

Exercise 5: Socket needed for simultaneous connections (2 pt)

1. ($\rightarrowtail$) The UDP server described in Section 2.8 needed only one socket, whereas the TCP server described in Section 2.7 needed two sockets. Why? If the TCP server were to support n simultaneous connections, each from a different client host, how many sockets would the TCP server need?

Exercise 6: HTTP and delay (6 pt) For each of the following http scenarios: calculate the number of round trip times (RTT) required to fetch and receive an index.html file with 5 embedded jpeg images (J1, ..., J5). Assume all the images and the index.html file are individually (but not together!) small enough to fit in one packet. Include the RTT for any TCP connection setup, but dont worry about the TCP connection closing time.

1. ($\curvearrowright$) http 1.0 with no parallel tcp connections

2. ($\curvearrowright$) http 1.0 using up to 4 parallel tcp connections (your solution should minimize the number of total RTTs)

3. ($\curvearrowright$) http 1.1 with persistent connections and no pipelining

4. ($\curvearrowright$) http 1.1 with persistent connections and pipelining

Exercise 7: Efficiency of Bit-Torrent (10 pt, including 1pt for the last)

Motivation: One important question in a torrent is how likely is that a peer has no information to request from its neighbors. This problem allows you to provide a mathematical argument justifying that it is unlikely.

Consider a torrent containing M peers and, without loss of generality, let Alice and Bob be two of them. We assume that the files to exchange contains N chunks. We also assume that, at the particular time t we are considering, the chunks that a peer has are chosen uniformly at random among the N chunks, and this choice is independent between two peers.

1. ($\curvearrowright$) Assuming that Alice has n_A chunks and Bob has n_B chunks, what is the probability that Alice has nothing to request from Bob?

2. ($\curvearrowright$) Assuming now that Alice's number of chunks is uniformly chosen in $\{0, 1, \ldots, N-1\}$, what is the probability that Alice has nothing to request from Bob (Bob still has n_B chunks)?

3. ($\curvearrowright$) Assuming now more generally that for any peer the number of chunks is chosen uniformly and independently in $\{0, 1, \ldots, N-1\}$, what is the probability that Alice has no information that could be of interest to her 5 neighbors?

4. ($\nrightarrow$) Assume that a peer has a single neighbor, and let $p(N)$ be the probability that this peer has nothing to download from its neighbor, which is a function of the number of chunks, N. Provide a simple function f such that:
$$p(N) \sim f(N) \text{ as } N \to \infty \quad .$$
What if each peer has k neighbors? Would you conclude that Bit-Torrent is efficient for large torrent with lots of chunks? Discuss the realism and limitations of the assumptions you made to answer that question.

NB: If that can help you to avoid any confusion, here is a mathematical formulation of the probability space that is considered.

The original file is divided into a set of N chunks that we denote by $\mathcal{N} = \{c_1, c_2, \ldots, c_N\}$. We consider a set $\mathcal{M}$ of M peers, that are indexed by i. Any peer $i \in \mathcal{M}$ has a random number of chunks X_i, which is an integer in $\{0, 1, \ldots, N-1\}$ chosen uniformly (in other words i has the same probability of having 0 chunks as having 1 chunks, the same probability to have 2 chunks etc.).

In addition, for any peer $i \in \mathcal{N}$, once you have chosen how many chunks this peer has, the actual set of chunks that it possesses is chosen randomly among all chunks in $\mathcal{N}$. In other word, there is a random subset of chunks (denoted by $Y_i \subseteq \mathcal{N}$) that this peer possesses and this subset is chosen among all subset with size X_i in $\mathcal{N}$, with the same probability.

In the first question, we ask you for the probability of this particular event (Alice has nothing she can request from Bob), conditionally on knowing that Alice has n_A chunks, and Bob has n_B. In other words, if you denote this event by E, we ask for: $\mathbb{P}\left[E | X_{Alice} = n_a, X_{Bob} = n_b\right]$. In the second question, we do not assume that we know in advance the number of chunks for Alice, so that the probability is $\mathbb{P}\left[E | X_{Bob} = n_b\right]$.

For more on conditioning, please refer to either Chapter V in *An Introduction to Probability Theory and Its Applications*, W. Feller, John Wiley (1967), or section 1.2 in *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, P. Bremaud, Springer (2000).