CSE P 590 SG Assignment #1

Due January 20, 2004

This assignment is due in class (before class begins) on Tuesday, January 20, 2004. You may bring it to class, or turn it in beforehand to the TA (electronic submissions may be emailed, or you can put it in my hand or my mailbox). Please do not turn in a late assignment.

For some of these problems, you will need to research, measure, or guesstimate various relevant design or system parameters. Always document assumptions or choices you make as appropriate, and cite your sources. If you need clarification of any question, ask. If you suffer an insight and want to take one of the questions in a new direction, feel free to run with it (though you might want to check that you've also got the basics down).

This is an individual assignment. You may discuss the questions and any broad ideas with your classmates, but you must do the work yourself and turn in your own work. For those questions that involve researching system parameters (e.g. 7), it is okay to share tips on where to find the necessary information, but do the research yourself, rather than sharing the actual numbers you discover, and be sure to cite any tips you do use. If you have any questions about these policies, feel free to ask either the professor or the TA.

There are seven parts to this assignment. Don't forget to turn the page over.

DNS Spelunking

Using existing DNS tools (such as `dig` on UNIX) to perform direct measurements, answer the following questions, and briefly explain how you ascertained your answers:

1. What is the complete list of authoritative name servers for the following domains? Can you offer any guesses as to why these domains have set up their nameservers as they have?
   (a) `.com`
   (b) `.edu`
   (c) `.ca`
   (d) `.cs.washington.edu`
   (e) `.yahoo.com`
2. Which of the name servers that you discovered are willing to serve recursive queries?
3. Which of the name servers that serve recursive queries (from 2) perform negative caching?
4. Try looking up non-existant `.cs.washington.edu` names using recursive lookup on the servers from 3. Is there a perceptible performance difference for names that are in the negative cache, versus those that are not? How significant do you think negative caching is for DNS performance?
5. Where does Andy's e-mail get delivered? Look up all the available records for `tioga.cs.washington.edu` (note that `dig`, by default, shows only the `A` records). What happens if one of the mailservers is down?

6. If you wanted to attack one of these name servers (meaning any that you found, not just those from parts 2 and 3), what attacks are possible and what would their effects be? To what extent do recursive query and/or negative caching support affect vulnerability to these attacks? (It should go without saying that you should **not** attempt to attack these servers in practice—you would likely get into serious trouble, and we would disavow any knowledge of your mission.)

7. Research (using your favorite method—mine is Google) how many `.com` registrations happen per day.

   Recall that we saw that a huge source of traffic to TLD servers is answering queries for bogus domain names. Let us consider a design where all DNS resolvers store the *complete* list of NS records for all second-level domain names (e.g. `yahoo.com`), and can therefore resolve all valid names *and* recognize all invalid names without contacting a TLD server (they would, of course, still need to talk to the second-level nameservers). To simplify the problem, we will consider only the `.com` domain (We can either imagine that we do the same thing for other top-level domains, or that we fall back on "normal" DNS for non-`.com` names; in either case we have much less data about the other top-level domains, so we won't worry about them.)

   Assuming that all second-level `.com` NS records have a two-day TTL (meaning that our new DNS must similarly ensure changes in second-level names are reflected within a two-day window), characterize the circumstances in which it would be cheaper (in terms of total network packets) to maintain and use this complete list of second-level names rather than interrogating the `.com` TLD server(s) directly.

   In order to answer this, you will likely need to fill out the definition of the algorithm somewhat, to define exactly what data is sent, by and to whom, and when. Is it possible to avoid transmitting the entire set of second-level NS records to all resolvers every two days? Don't go overboard in specification here; we aren't going to implement this system, we only want to figure out when it might be better.