Carnegie Mellon
Computer Science Department.
15-744 Fall 2009 Problem Set 3

This problem set has 13 questions. Answer them as clearly and concisely as possible. You may discuss ideas with others in the class, but your solutions and writeup must be your own. If you do discuss at length with others, please mention in your solution for the problem who you collaborated with. Do not look at anyone else's solutions or copy them from anywhere.

This assignment is due by **5:00pm, Wednesday, Nov 25th** either in class or to the course secretary in Gates 9118.

Glossary

BGP: The Border Gateway Protocol
DHT: Distributed Hash Table (like Chord)
DNS: The Domain Name System
TTL: Time To Live values (how long a DNS record may be cached)
NS Records: Name Server records; those records that point to more specific authoritative servers for a DNS name.
RIAA: The Recording Industry Association of America. Defn (1) An industry group that attempts to ensure that artists are fairly compensated for their work. Alternate meaning: (2) An industry group that believes that launching a campaign of fear and intimidation by indiscriminately suing its customers and seven year old children is the most effective way to boost sales and ensure goodwill.

Solution: Total: 100 points.

A DNS Tools

In the following questions, you will learn to use two useful tools for querying for Domain Name System (DNS) information: `nslookup` and `dig`.
The `nslookup` program queries Internet domain name servers (DNS). Entering the command nslookup will give you the name of the name server your system knows and its IP address. The list of name servers used by a Unix machine can usually be found by looking at the file */etc/resolv.conf*. Read the man page for nslookup and answer the following questions.

1. What is the IP address of the school's web server (`www.cs.cmu.edu`)?

   Solution: 5 points. 128.2.203.164

2. When you send mail to `somebody@steelers.com`, which machine does the mail go to? What are the machines used to process mail sent to `somebody@cmu.edu`?

   Solution: 5 points. `somebody@steelers.com` - `smtp-in.load.com` (209.58.232.24)
   `somebody@cmu.edu` - `CMU-MX{1,2,3}.andrew.cmu.edu` (128.2.11.1, 128.2.32.46, 128.2.10.160)

`dig` is another program that allows you to query DNS servers. For the purpose of this question, you should use the following format to invoke dig

```
dig +norecurse @<name.of.dns.server> <record-type><domain-name>
```

where

- `<name.of.dns.server>` is the hostname of the DNS server you wish to query such as `A.ROOT-SERVERS.NET`
- `<record-type>` is the type of DNS record you wish to retrieve, such as ANY, MX, etc.
- `<domain-name>` is the name of the host or domain you seek information on.

The DNS is a distributed architecture that uses hierarchical delegation. At the top of the system are the root name servers, which know which DNS server is responsible for each second-level domain (such as `cmu.edu`). If you send a root server a query for a particular machine, you will receive a reply listing the servers that have been delegated authority for that machine's second-level domain. It is common for a large domain such as `cmu.edu` to further delegate to departmental or workgroup DNS servers, which you discover by querying the second level servers.

In order to discover the chain of delegation in use at Akamai, run a series of NS queries for `a1793.x.akamai.net`. You may wish to start with any of the 13 root servers (`[a-m].root-servers.net`), and you should continue your sequence of queries until you stop getting new delegations (in some domains this is indicated by a DNS server returning you a delegation pointing to itself, and in other domains this is indicated by a DNS server returning you a SOA record instead).

As an example here is the delegation chain for `aol.com`:

Server queried	NS delegates to
`B.ROOT-SERVERS.NET`	`A.GTLD-SERVERS.NET`
`A.GTLD-SERVERS.NET`	`DNS-01.NS.AOL.COM`, `DNS-0[267].NS.AOL.COM`
`DNS-01.NS.AOL.COM`	`DNS-01.NS.AOL.COM`

3. Generate the delegation chain for `a1793.x.akamai.net`. Present your results in the table form shown above. Each NS query will typically return two or more answers: choose among them at random. If you query a server and get a timeout, choose an alternate server.

Solution: 5 points.

Server queried	NS delegates to
{A-M}.ROOT-SERVERS.NET	{A-M}.GTLD-SERVERS.NET
{A-M}.GTLD-SERVERS.NET	z{a-h}.akamaitech.net
z{a-h}.akamaitech.net	n{0-7}x.akamai.net
n{0-7}x.akamai.net	returns A record

DNS is also used for reverse lookup, i.e. to translate IP addresses into hostnames. Again, the database is distributed in a hierarchical fashion, with a wrinkle. The most-specific part of a domain name is on the left (i.e. `ux1` in `ux1.sp.cs.cmu.edu`), but the reverse is true of IP addresses (i.e. in 128.2.198.101, 128 is top-level, 128.2 is Carnegie Mellon in general and 128.2.198 belongs to the Computer Science Department. Thus, address-to-name mappings are discovered by reversing the bytes of the IP address and making queries in a special domain. To turn 128.2.198.101 into a hostname, various servers are sent queries seeking PTR records for `101.198.2.128.in-addr.arpa`. The first query would be:

```
dig @a.root-servers.net PTR 101.198.2.128.in-addr.arpa
```

You will know when you're done when your query gives you back a PTR record in addition to (or instead of) NS record.

Note that you have to reverse the bytes in the address, i.e. start with the lowest level byte, e.g.,

```
dig @a.root-servers.net PTR 36.0.26.18.in-addr.arpa
```

Server Queried	NS delegates to
A.ROOT-SERVERS.NET	STRAWB.MIT.EDU
STRAWB.MIT.EDU	returns the PTR record

The hostname is `mintaka.lcs.mit.edu`.

4. Fill in a table like the one above showing a query chain for the IP address 128.2.198.61.

Solution: 5 points.

Server Queried	NS delegates to
{A-M}.ROOT-SERVERS.NET	{basil,chia,dill,epazote,figwort,henna}.ARIN.NET
*.ARIN.NET	{t-ns2-sec,t-ns1}.net.cmu.edu, cabbage.srv.cs.cmu.edu
*.net.cmu.edu	{SPINACH,CABBAGE,LETTUCE}.SRV.CS.CMU.EDU
*.SRV.CS.CMU.EDU	returns the PTR record

The third step is skipped if you asked `cabbage` directly after the second step.

B DNS Redirection

Harry Bovik is working on a web site that has multiple replicated servers located throughout the Internet. He plans on using DNS to help direct clients to their nearest server replica. He comes up with a hierarchical scheme. Harry has divided his server replicas into three groups (east, west and central) based on their physical location. A typical query occurs as follows:

- When a client makes a query for `www.distributed.hb.com`, the root and `.com` name servers are contacted first. It returns the name server (NS) record for `ns1.hb.com`. The TTL of this record is set to 1 day.
- The `ns1.hb.com` name server is then queried for the address. It examines the source of the name query and returns a NS record for one of {`east-ns, central-ns, west-ns`}`.distributed.com`. The choice of which name server is based on where `ns1` thinks the query came from.
- Finally, one of {`east-ns, central-ns, west-ns`}`.distributed.com`. is contacted and it returns an address (A) record for the most lightly loaded server in its region.

Answer the following 3 questions based on this design.

5. Harry's name server software has only two choices for TTL settings for A and NS records - 1 day and 1 minute. Harry chooses the following TTLs for each record below:

   1. NS record for {`east-ns, central-ns, west-ns`}`.distributed.com` - 1 day TTL.
   2. A record for {`east-ns, central-ns, west-ns`}`.distributed.com` - 1 day TTL.
   3. A record returned for the actual Web server - 1 minute TTL.

   Briefly explain why Harry's choices are reasonable, or why you would have made different choices.

Solution: 4 points. The name server for a client is based on the region and hence probably does not change very often. Therefore, Harry sets the NS and A records for the name server address to 1 day.

Harry wants the name server to direct clients to lightly loaded web servers. To do this, Harry must be able to control which web servers each client goes to. Therefore, Harry sets the TTL of the web server A record to 1 minute, so that clients won't cache the record for very long and will ask the name server which web server to use for subsequent requests. If the TTL was 1 day, each client would cache the first A record it got and then continue using the same web server for the entire day even if it becomes overloaded.

6. In general, name resolution systems map names based on the name and context. In this particular case, what are ***TWO*** items of context that the name resolution uses?

Solution: 5pts

1) the IP address of the local name server

2) the load on the servers in the region

7. Harry's Web site is especially popular among CMU students. The CMU network administrator estimates that there is one access from CMU every 3 minutes. Each access results in the application resolving the name `www.distributed.hb.com`. Assume the following:

   - No other DNS queries are made in CMU
   - All CMU clients use the same local name server.
   - This local name server is mapped to the `east-ns` region.
   - Web browsers do not do any caching on their own.

   How many accesses per hour will be made to the following name servers to resolve these CMU queries? Explain your calculation.

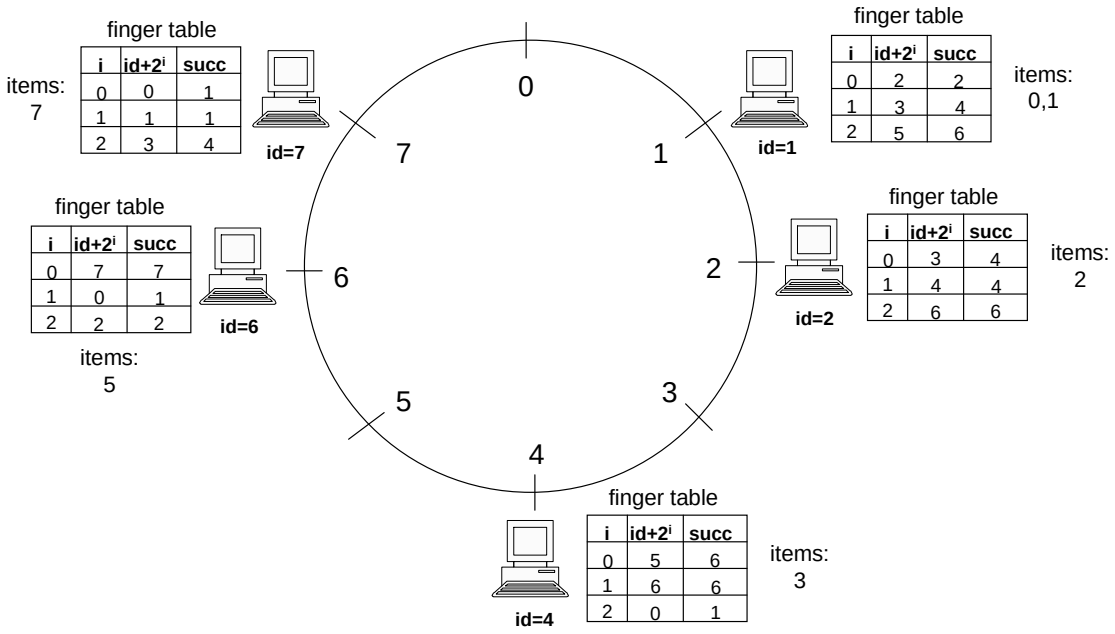
   1. The Root Servers
   2. `ns1.hb.com`
   3. `east-ns.distributed.com`

Solution: 5 points.

1. The Root Servers - 1/24 requests/hour
2. `ns1.hb.com` - 1/24 requests/hour
3. `east-ns.distributed.com` - 20 requests/hour

C DHTs

8. Dave, in fear that the RIAA will shut down his centralized P2P server (like Napster), sets up a Chord DHT for lookups and routing in his peer to peer network. Unfortunately (or fortunately, for you), Dave's P2P network is not very popular and only consists of five peers at the moment with finger tables and items illustrated below. For example, *node 4* has *item 3*.



(a) List the nodes that will receive a query from *node 2* for *item 0*.

> **Solution: 6 points.** $2 \to 6 \to 7 \to 1$. Node 6 can not contact node 1 directly (even though it has a finger to node 1) because $1 \notin (6, 0]$. Chord finds the *predecessor* of 0 first and then finds the item on the immediate successor of the predecessor. This is to ensure that we don't "skip over" the successor when our finger table is stale.

(b) Suppose node 4 crashes. *node 7* queries for *item 5*. List the nodes that will receive this query, assuming the the tables have had time to converge after noticing that node 4 has left.

Solution: 6 points. $7 \to 1 \to 2 \to 6$

D Multi-Homing Multi Homes

This question is intended as a "back of the envelope" analysis of the sort that researchers do every day when thinking about systems; it's likely there is no perfect answer, but please justify briefly the numbers and sources of data you use as a basis for answering this question.

9. After reading about the great improvements that one can achieve using multi-homing [Akella et al.; Andersen et al.; Sariou et al.; etc.], you find yourself wondering: Why can't I multi-home my house? It seems like a great way to get twice the bandwidth and much more than twice the availability for only twice the cost. And if you were multi-homed, you wouldn't have an outage if a large local Internet service provider started having problems paying the bill for its fiber. Attractive all around!

   So, let's look at multi-homing by the numbers. Come up with a back of the envelope calculation about what would happen if every broadband Internet-connected household in the United States used BGP multi-homing.

   (a) How many routing table entries would this require in the core, assuming no aggregation? Note that several factors contribute to routing table size, such as the number of route entries, the number of peer routers that a router has, etc. State what assumptions you're making behind your calculation.

   Solution: 5 points. According to the Pew Internet & American Life Project [1], there are 84 million broadband users in the united states. A backbone router is likely to have only a small number of full-table peers; for simplicity, let's assume that it's 10. As a result, the BGP routing table would take $84M \cdot 10$ entries, and the forwarding table would require $84M$ entries (remember that only the best path is chosen to put in the forwarding plane). That's about 100x more than the largest routers can handle today. Interestingly, that's *only* 100x more.

   [1] `http://www.pewinternet.org`

   (b) How much memory would you estimate that a core router would need for its routing table with and without this new glut of multi-homing? (Hint: For current table size information, google for the "CIDR report")

   Solution: 5 points. The current routing table has about 220,000 entries. Assuming again 10 peers per core router, today we need enough memory for 2.2M entries. A BGP route entry contains a number of attributes:

Attribute	Size (bytes)
Prefix	4
Prefix Len	1
Peer AS	2
next-hop	4
AS Path	Variable string of 16-bit ints. say 16 bytes
Community	Variable string of 32-bit ints, say 16 bytes
localpref	4
origin	1
MED	4

   Each BGP table entry will take roughly 50 bytes of memory. Thus, one might expect a full table in a core router today to require about 128MB of DRAM, rounding up to a convenient number.

   With every broadband user having a table entry, this number would grow to about 40GB of DRAM. (The forwarding table would take 4GB of very fast memory, which is much more troublesome.)

(c) Picking a reasonable value (justify it) for the frequency of routing updates, how much bandwidth do you think this would require at each BGP-speaking router that carries a full routing table? (Don't make this complicated worrying about BGP update dynamics. Assume that a single link change or router reboot only results in one message being propagated; we can mentally multiply later.)

Solution: 5 points. In 1999, Labovitz et al. observed "several hundred thousand" routing updates per day at the Mae-East exchange; at the time, the routing tables contained about 50,000 entries. This suggests at most a few updates per day per prefix [1]. However, Rexford noted in 2002 that larger and more popular destinations are more stable, and that most routing update traffic comes from smaller prefixes [2]. Given the relative unreliability of home Internet connections, we might expect a higher rate; perhaps 10 per day.

At 10 updates per day and roughly 50 bytes per update for 84M prefixes from 10 peer routers, we might expect the routing updates to consume about 390 gigabytes of bandwidth between each router pair. That might be a bit high an estimate, but it's probably within a factor of five or so.

[1] "Origins of Internet Routing Instability", Craig Labovitz, G. Robert Malan, and Farnam Jahanian.
[2] "BGP Routing Stability of Popular Destinations", Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang, Intrnet Measurement Conference 2002.

10. One way of taking advantage of multi-homing is to use your upstream Internet links in parallel in various ways. Ignoring little (important) details like actually making TCP work, if you used a scheme that sent a copy of your data down each of your links in parallel:

(a) How available might you expect your system to be if the links fail independently with probability p? If each link has roughly 99.5% uptime (a number that sounds good, but which equites to 1.85 *days* of downtime per year), how much uptime would your combined system have? (There is nothing tricky about this question. It's mostly to give context and some concrete numbers for the next question.)

Solution: 5 points. You'd expect it to have $1-((1-p)^2)$ availability, with $(1-p)^2$ downtime per year. That amounts of 99.9975% availability and 13.14 minutes of downtime per year.

(b) You suspect that in practice, the achieved end-to-end availability—the ability of the multi-homed source to reach a particular destination when it wants to—will be lower than this theoretical limit, because there will be correlations in the failures across the two links. List, *in order*, what you think the three most likely causes of such correlated failures will be, and *briefly* (one or two sentences) justify your decision.

Solution: 5 points.

1. **Destination failures.** If the host is trying to reach a less reliable destination, it doesn't matter how good the host is. Both sides must be up to communicate.
2. **Power failures**. Even a simple brownout or power "burp" would be enough to disrupt service for a few minutes.
3. **Non-independent link failures** such as fibers that run through the same physical conduit.

Other causes include **failures that BGP cannot mask**, such as configuration errors or many software/hardware bugs; **BGP convergence time** - when a link fails, it may take several minutes for BGP to recover, and 13 minutes of downtime per year does not allow for much of that.