Carnegie Mellon
Computer Science Department.
15-744 Spring 2007 Theory Problem Set 2

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, April 4th** either in class or to the course secretary in Wean Hall 8018.

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.
ExOR: Extremely Opportunistic Routing, a routing protocol from the Roofnet project that attempts to route packets based upon which set of intermediates *actually* received the packets. **DSR**: Dynamic Source Routing, an ad-hoc mobile routing protocol
DSDV: Destination Sequenced Distance Vector. As above.
ETX: Expected Transmission Count, an ad hoc routing path quality metric

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`)?
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`?

`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.

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.

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.

6. Harry notices that many clients are directed to servers in the wrong region (i.e. the client is not in the same region as the Web server chosen). He tracks down the problem and sees that the clients appear to be in some other region during name resolution. Why is this happening?

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:

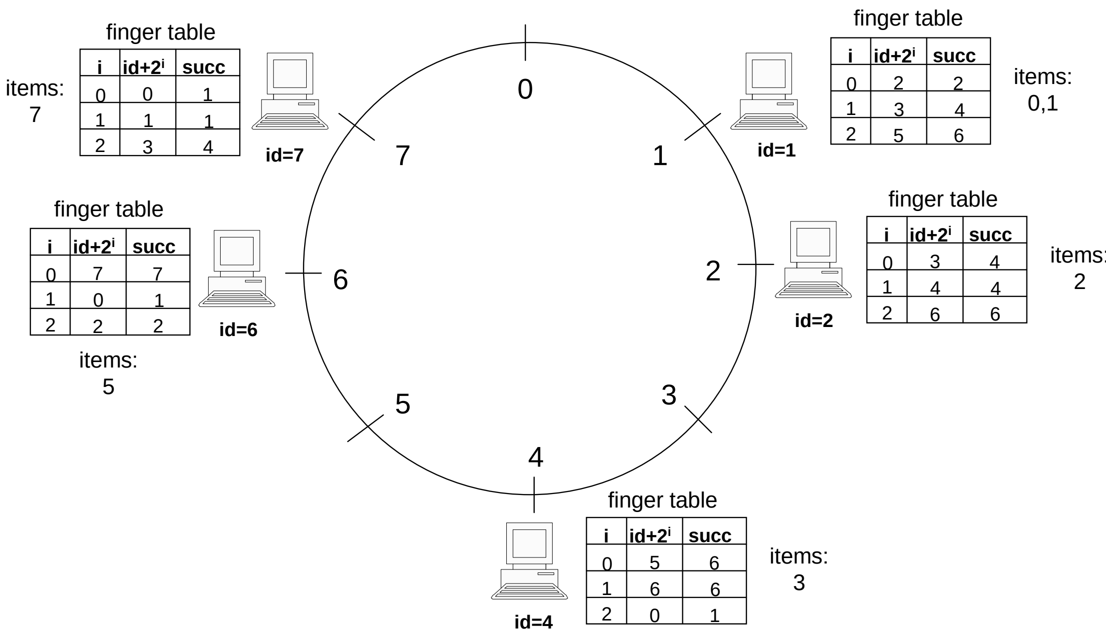
   - No other DNS queries are made in CMU
   - All CMU clients use the same local name server.
   - This local name server is mapped to the `east-ns` region.
   - Web browsers do not do any caching on their own.

   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`

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*.

(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.

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.

(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")

(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.)

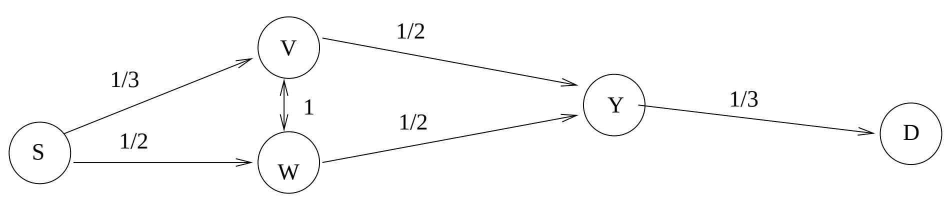
10. One way of taking advantage of multi-homing is to use your upstream Internet links in parallel in various ways. Ignoring little (important) details like actually making TCP work, if you used a scheme that sent a copy of your data down each of your links in parallel:

(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.)

(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.

E ETX-OR

Consider the wireless network pictured below. Assume that links experience Bernoulli losses. The labeled edges indicate the *combined* delivery ratio (i.e., the probability that a packet is successfully received in the forward direction and that the acknowledgement is received in the reverse direction). V and W can hear each other perfectly. If there is no edge, assume that no packets make it through.



11. Assume that the link layer performs retransmissions. What is the expected number of transmissions to send one packet from S to D using the ETX metric along the path S-W-Y-D?

12. What is the expected number of transmissions needed to send a packet from S to D using ExOR? (Assume that there are enough packets in the batch so that the overhead of ExOR headers and its batch maps is insignificant, and that batch maps are received 100% reliably.)

13. Geographic routing has been proposed as an alternative to ad hoc routing protocols such as DSR and DSDV. In geographic routing, nodes are "identified" by their geographic address (e.g., their GPS coordinates), and intermediate nodes forward packets greedily in the direction of the recipient.[1] These schemes scale better than DSR or DSDV because they do not need to propagate information about nodes' locations globally—they just propagate them within a local area and rely on geographic forwarding to get packets near the receiver. They do, however, face the problem of requiring a way to map the "real" ID of a node to its geographic coordinates, particularly if nodes are mobile.

    After thinking about the problem for a while, you have a hunch that you could combine geographic routing with some of the consistent hashing-like techniques you've learned about in 15-744 to create a scalable ad-hoc routing system. Your idea is to use consistent hashing on the node's real ID to map to a geographic location such that every node publishes its ID "towards" that location, and queries for that node are also routed towards that location.

    Make some simple assumptions about the mesh:

    - It contains N nodes
    - The nodes in your network are deployed within a square area between (0,0) and (max_x, max_y).
    - The nodes are uniformly distributed
    - Nodes forward packets by sending to the immediate neighbor in the direction of the destination. There's no fancy hop-over magic going on.
    - The mesh is sufficiently dense that queries don't end up in a hole.

    (a) Define a function that you could use to map from a human-readable name (like a DNS name – `foo.wireless.net`) to a geographic coordinate within the scope of your network. Start by using the SHA-1 hash of the human readable name to convert it to a 160 bit number.

    (b) How many packet transmissions would it take in expectation for every node to transmit its geographic identifier to the location handling its name mapping and for every node to send one packet to one randomly chosen node? Each node choses a destination node at random. Assume that there is no caching along the path. A packet transmission is one packet being transmitted by one node; it may take multiple transmissions to reach its destination.

    (c) *Assuming no route caching*, how many packet transmissions will it take for every source to send one packet to a randomly chosen destination using DSR?

    (d) **Thought question, NOT required, and the course staff has not sat down to figure out if the question is tractable!** How many packet transmissions would it take with DSR in the same network *with* route caching, assuming no mobility?

[1] Such protocols must provide a way to route around "holes" in the topology; one such mechanism is the one described in a paper by Y.J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic Routing Made Practical."