15-744 Computer Networks (Fall 2011)
Homework 3

Due: Nov. 7th, 2011, 3:00PM (in class)

Name:
Andrew ID:

November 11, 2011

A BitTorrent and DHT

BitTorrent is the most popular peer-to-peer file sharing system today. In the original design of BitTorrent system, there is centralized component called "tracker", which informs each peer the presence of other peers in the same torrent. More specifically, to join a torrent a peer needs to first connect to the tracker listed in the .torrent file. Then the tracker replies this new peer with a list of known peers currently downloading the same torrent so that this peer can connect to them. Each tracker is usually serving a large number of torrents.

1. Briefly state the advantages and disadvantages of using tracker to manage peers

   Solution: pro: easy to deploy, easy to manage peers, easy to apply optimization
   con: single point of failure

2. Briefly, how could BitTorrent system use a DHT such as Chord to replace the tracker and maintain a directory of many torrents?

   Solution: Chord could be used to build a dictionary, where a hash of the torrent can be used as a key, and the value associated with that key is simply the list of peers in the torrent currently.

3. Your DHT-based tracker is more and more popular and growing much bigger. Meanwhile you found that it takes a long time to perform lookups. Explain why are lookups taking longer.

   Solution:

   Chord requires up to $\log n$ messages to complete a lookups, and when n gets to be in the tens of thousands, this is ten to fifteen lookups.

   Second, there's no locality in chord, so when the users are located all over the world, each message may have to travel a very long distance.

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.

4. 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.

5. 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

6. 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 Hashing and Caching

Bovik is trying to figure out a scheme his clients should use, so that given a URL, they can find the appropriate CDN node to fetch the content from.

Bovik has come up with a hash function h that takes a string and maps it to a real number in the range $[0, 1)$. Assume there are 3 CDN nodes with names such that $h(node_1) = 0.1, h(node_2) = 0.85, h(node_3) = 0.5$. When a client needs to fetch a URL and has to decide which replica to query, it picks $node_i$, such that the absolute value of the difference between $h(node_i)$ and $h(URL)$ is minimum. This scheme does *not* use circular mapping - it's just numeric closeness. This technique is **"scheme 1"**

7. Assuming all URLs are equi-popular, which node is likely to see the highest load?

> **Solution:** 0-.3 go to node1, .3-.675 go to node3 and .675 - 1 go to node2. This makes $node_3$ the most loaded.
>
> If you mapped it in a circular fashion, $node_1$ gets .975 - .3, $node_3$ still gets .3-.675 and $node_2$ gets .675 to .975. $node_3$ is still the most loaded.

Since load can be unevenly distributed in the above scheme, Bovik is not satisfied with the scheme. Instead, he thinks of a new arrangement. Let there be m CDN nodes in all; sort them using the $h(node_i)$ values. If the rank of a node is r, $(0 \leq r \leq m - 1)$, it is responsible for storing all URLs that map to the interval $[r/m, (r + 1)/m)$. This new scheme is called **"scheme 2"**.

8. Why might a CDN with a large number of nodes (that occasionally crash and are later repaired) choose scheme 1 over scheme 2?

> **Solution:** Scheme 2 forces you to frequently move content between the nodes. Scheme 1 only forces the two neighboring ranges to adjust. This is one of the important properties of consistent hashes.