CS425 Fall 2019 – Homework 1

(a.k.a. "All the Presidential Candidates' Interns")

Out: Aug 27, 2019. Due: Sep 24, 2019 (Start of Lecture, 2 pm Central)

Topics: Clouds, Mapreduce, Gossip, Failure detectors, Membership, Grids, P2P Systems (Lectures 1-8)

Instructions:

1. **Attempt any 8 out of the 10 problems** in this homework (regardless of how many credits you're taking the course for). If you attempt more, we will grade only the first 8 solutions that appear in your homework (and ignore the rest). Choose wisely!
2. Please hand in **solutions that are typed** (you may use your favorite word processor. We will not accept handwritten solutions. Figures and equations (if any) may be drawn by hand (and scanned).
3. **All students (On-campus and Online/Coursera)** – Please submit PDF only! Please submit on Gradescope. [https://www.gradescope.com/]
4. Please **start each problem on a fresh page**, and **type your name at the top of each page**.
5. Homeworks will be **due at the beginning of class on the day of the deadline. No extensions. For DRES students only:** once the solutions are posted (typically a few hours after the HW is due), subsequent submissions will get a zero**. All non-DRES students must submit by the deadline time+date.**
6. Each problem has the same grade value as the others (10 points each).
7. Unless otherwise specified, the only resources you can avail of in your HWs are the provided course materials (slides, textbooks, etc.), and communication with instructor/TA via discussion forum and e-mail.
8. You can discuss lecture concepts and the questions on Piazza and with your friends, but you cannot discuss solutions or ideas. All work must be your own.

Prologue: In the year 2020 AD, the US will hold Presidential Elections. Most Presidential Campaigns today run distributed systems and cloud computing for storage and analytics of data such as the campaign, events, voters, schedules, etc., (e.g., both the Obama and Romney campaigns in 2012 built and used distributed systems, including cloud computing and mobile computing).

This homework uses fictitious stories and characters from a few ongoing presidential campaigns to frame the homework problems. The choice of candidate names below is purely arbitrary, and nothing should be interpreted by inclusion omission of a campaign. Any resemblance to persons, places, or events, living or dead, past, present or future, is purely coincidental. These stories and this homework are not intended to side with or against any candidate, or make comments about any candidate. They are not at supporting or endorsing, nor at criticizing or disparaging, any candidate, campaign, people in campaigns, political parties or affiliations, or voters or citizens or persons living in the US. All interns are fictional, as are their goals and actions.

<u>Problems</u>:

1. An intern in Kamala Harris' campaign wants to write a Mapreduce program. In MapReduce, one writes a program for Map that processes one input line at a time and outputs zero or more (key, value) pairs; and one writes a program for Reduce that processes an input of (key, all values for key). The iteration over input lines is done automatically by the MapReduce framework. The intern would like to know who are the influential Twitter users most similar to the candidate, and would like to use Hadoop for this. The intern uses an input file containing information from Twitter (which is an asymmetrical social network) about which users "follow" which other users. If user a follows b, the entry line is (a, b) – you can assume this data is already sharded HDFS and can be loaded from there. Can you help the intern? Write a MapReduce program (Map and Reduce separately) that outputs the list of all users U who satisfy the following three conditions simultaneously: U has at least a million followers, and U herself/himself follows at least 10 users, and U follows at least one user V who in turn has at least 2 million followers (e.g., @KamalaHarris would be such a U). You can chain Mapreduces if you want (but only if you must, and even then, only the least number). Your program must be as highly parallelizable as possible.
2. An intern in Beto O'Rourke's campaign wants to write a MapReduce program (Map and Reduce separately) that outputs all "triples", i.e., sets of users (a,b,c) such that a follows b, b follows c, and c follows a (there might be other follows-relationships among the triple). You can chain Mapreduces if you want (but only if you must, and even then, only the least number). Be sure to output each triple at most once (e.g., in sorted order). You don't need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the "key" in Map output.

3. An intern in Bernie Sanders' Campaign has designed a failure detection protocol that is a modified ring failure detection protocol. It works as follows: each process *i* selects *k* other processes at random and asks these *k* processes to send it (*i*) heartbeats. Heartbeats are not relayed (so this is not gossip, but more like ring failure detection), and process *i* times out if it doesn't receive heartbeats. Heartbeat targets are selected at the protocol start and are not changed thereafter. A process is detected as failed if any of its heartbeat receivers do not receive expected heartbeats within a pre-specified timeout.
    a. The intern believes that this algorithm provides completeness up to *k* failures. Are they right? If yes, argue why (informal proof). If no, give a counter-example, and also state what completeness the algorithm does provide.
    a. The intern also claims this algorithm satisfies accuracy. Are they right? If yes, argue why (informal proof). If no, give a counter-example, and also state what kind of accuracy the algorithm does provide.
4. An intern in Elizabeth Warren's campaign is designing a membership protocol for a system of N nodes where you are guaranteed to not to have more than N/4 failures (no new processes join; processes only fail). What is the *least* amount of memory a heartbeat-based membership protocol can use so that it satisfies completeness? You can state memory in terms of number of entries. Prove that your algorithm has completeness.
5. An intern in the Andrew Yang campaign wants to use gossip to spread the word about the candidate. Due to limited resources, the intern decides to use only a partial membership list for the gossip. The membership list at each process is selected uniformly at random across the entire group (somehow, the messages take care of it – don't worry about the protocol part). Processes don't fail or join. Each message is gossiped to *m* randomly selected neighbors (from the membership list), where $m < k$, and m = O(log(N)), with the latter needed to ensure spread of gossip. The intern argues that due to random selection, the overall "behavior" of this protocol (in terms of dissemination time of gossips, etc.) is the same as in the case where all processes might have had full membership lists (known everyone in the group), and each gossip was sent to *m* neighbors. Is the intern right? If yes, then give a proof. If no, show why.
6. An intern in the Tulsi Gabbard campaign wants to connect all potential fans of the candidate via a p2p system. But the intern wants to modify the Chord DHT rules to make it topologically aware of the underlying network latencies (like Pastry is). Design a variant of Chord that is topology-aware and yet preserves the O(log(N)) lookup cost and O(log(N)) memory cost. Use examples or pseudocode – whatever you chose, *be clear*! Make the least changes possible. You should only

change the finger selection algorithm to select "nearby" neighbors, but without changing the routing algorithm. Show that (formal proof or informal argument):

   a. Lookup cost is O(log(N)) hops.
   b. Memory cost is O(log(N)).
   c. The algorithm is significantly more topologically aware than Chord, and almost as topology aware as Pastry.

7. An intern in the Joe Biden campaign wants to reach all supporters, and so designs a new P2P DHT called "AHDHT" (A Hop DHT) where all peers know the addresses of all peers.
   a. How would you build this design so that any file can be fetched from any peer in just 1 hop?
   b. Someone proposes sending all join/leave/fail requests immediately to all other processes in the DHT, in order to keep all membership lists always fresh (thus all lookups would be correct). Given your design from (a), and a churn rate of 100% per hour, what is the amount of bandwidth necessary to keep all membership lists up to date in a system containing 1 million peers? (That is, we want each membership change to be instantly reflected in all membership lists). Is this bandwidth tenable in a Wide Area Network today?
   c. You decide to use gossip-based membership, where each entry has an IPv6 address, a port number, a gossip count (int with 4 bytes), and a last updated time (UTC time). You have only 2 GB to store the membership list. How large can the P2P system be in size (assuming no memory waste or extra memory overhead)?
8. In order to coordinate all the other interns in the Amy Klobuchar campaign, an intern decides to use a peer to peer system. The intern uses the Gnutella system. At one point of time, the Gnutella topology looks like a virtual ring with 11 processes, with each process connected to its immediate two clockwise neighbors and immediate two anticlockwise neighbors (thus each process has four neighbors). All links are bidirectional. Answer three parts:
   a. A process sends a Query message with TTL=5. How many of the processes receive this Query message (include the sender)?
   b. What is the minimum TTL in a Query message to ensure all nodes in the system receive the Query (no matter the timing of the Query forwarding)?
   c. If we add a 12th process (the candidate) that is connected to all the 11 processes, what is the minimum TTL in a Query message to ensure all nodes in the system receive the Query?
9. An intern in the Donald Trump campaign is playing around with a Chord DHT ring using m = 12, nodes with the following peer ids (or node ids) having joined

the system in this order: 2020, 2016, 2012, 2008, 2004, 2000, 1996, 1992, 1988, 1984, 1980 (these are all US presidential election years!). Then, answer the following questions:

   a. Show or list all finger table entries for node 1980.
   b. When all finger tables and successors have converged, show the path taken by a search (or query) message originating from node 1980 intended for the key 2020.
   c. Node 2012 fails. List all the nodes whose finger tables need to be updated.

10. An intern in the Kirsten Gillibrand campaign is trying to decide between Pastry and Kelips. The estimated size of the system will be 1 million nodes. Calculate, for each of Pastry and Kelips, the following metrics and say which of the two systems is better:
    a. Memory per node, in KB (for neighbors only, assuming 20 Bytes per neighbor entry).
    b. Failure detection bandwidth per node, focusing on only background bandwidth in the steady-state (Pastry uses heartbeats to 1 hop neighbors). You can give the bandwidth in messages per second or KBps (if you do the latter, state your assumptions on unit/message sizes).
    c. Lookup latency, in number of hops.