CS425 Fall 2016 – Homework 1
(a.k.a. “Friday Night Plights”)

Out: Aug 22, 2016. Due: Sep 20, 2016 (Start of Lecture)

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 hardcopy 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.
3. Please start each problem on a fresh sheet (not just page), and type your name at the top of each sheet.
4. Homeworks will be due at the beginning of class on the day of the deadline. No extensions.
5. Each problem has the same grade value as the others (10 points each).
6. 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.
7. 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: Today, almost all professional sports teams (e.g., NFL, NBA, MLB, NHL, MLS, etc.) use distributed computing to analyze games, give feedback to players (sometimes in real time), and to decide whom to trade and when.

This homework uses fictitious stories and characters from sports teams to frame the homework problems. Any resemblance to persons, places, things, or events, living or dead, past, present, or future, is purely coincidental. These stories and this homework is aimed neither at endorsing, nor criticizing, any league, any sports team or persons associated with these teams or leagues.
**Problems:**

1. The Chicago Bears would like to increase their Twitter presence. As a warm up exercise they wish to find pairs of interested users on social media. They would like to use MapReduce for this. In MapReduce, one writes a program for Map that processes one input line at a time and outputs a (key, value) or nothing; 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. You are given an input file containing information from an asymmetrical social network (e.g., twitter) about which users “follow” which other users. If user a follows b, the entry line is (a, b). Write a MapReduce program (Map and Reduce separately) that outputs all pairs of users (a, b) who follow each other. You don’t need to write code – pseudocode is fine as long as it is understandable. Hint: Think about the “key” in Map output.

2. The Green Bay Packers laugh at the Chicago Bears and decide to one-up them by finding triples (instead of pairs). You are given an input file containing information from an asymmetrical social network (e.g., twitter) about which users “follow” which other users. If user a follows b, the entry line is (a, b). 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. But LeBron James snickers at both the above teams and says he by himself is more popular than either of those teams (it’s true! @KingJames has 32 M Twitter followers!). King James would like to know who are the Twitter users most similar to him, and would like to use Hadoop for this. There is 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. 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 a million followers (e.g., @KingJames satisfies this). 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.
4. The NE Patriots team has had too many injuries, so they decide to design a failure detector. A cornerback 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 directly. 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.

   a. The quarterback (Tom Brady himself) claims that this algorithm provides completeness up to $(k-1)$ failures. Is he right? If yes, argue why (informal proof). If no, give a counter-example, and also state what completeness the algorithm does provide.

   b. The running back (of the day) in the team 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.

5. The Chicago White Sox have $N$ players (processes) where you are guaranteed to not to have more than $N/3$ failures (no new processes join; processes just fail). What is the least amount of memory a heartbeat-based membership protocol can use so that it satisfies completeness? Prove that your algorithm has completeness.

6. The Chicago Bulls decide that like they don’t need Derrick Rose, gossip protocols don’t need all processes either in their membership lists. So they decide to build a membership protocol (for a system of $N$ processes) where each process has a membership list of size $k$. 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). 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 new Bulls’ point guard 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 he right? If yes, then give a proof. If no, show why.

7. The new manager and coach of the LA Lakers team feel the rest of country is too disconnected from their team (after Kobe’s retirement). He wants to connect all Lakers fans together using a P2P system, and have it be topologically aware. 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. Hint: Try to change only the finger selection algorithm, but not the routing. 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.

8. Someone smart on the Pittsburgh Penguins NHL team (after all, they won the Stanley Cup in 2016!) 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?

9. In order to have their players share team videos with each other (but not during the game, because that would be illegal!), the Indianapolis Colts decide to use a peer to peer system. They use the Gnutella system. At one point of time, the Gnutella topology looks like a virtual ring with 12 processes, with each process connected to its immediate three clockwise neighbors and immediate three anticlockwise neighbors (thus each process has six neighbors). All links are bidirectional. Answer three parts:

a. A process sends a Query message with TTL=3. How many of the processes receive this Query message?

b. What is the minimum TTL in a Query message to ensure all nodes in the system receive the Query?

c. If we add a $13^{th}$ process (the coach) that is connected to all the 12 processes, what is the minimum TTL in a Query message to ensure all nodes in the system receive the Query?

10. In order to win their next championship (after the century-plus drought) the Chicago Cubs realize they need to stop losing fans. They decide that they need to get their fans excited. They decide to use a Chord-like peer to peer system to connect the mobile phones of their fans with each other. The players also start
using this system – after a particular game, the players’ mobile phones are in a Chord ring using \( m = 12 \), nodes with the following peer ids (or node ids) join the system: 1908, 1910, 1920, 1930, 1940, 2000, 2014, 2015, 2016. Then, answer the following questions:

a. Show or list all finger table entries for node 1910.

b. When all finger tables and successors have converged, show the path taken by a search (or query) message originating from node 1908 intended for the key 2015.

c. Node 2015 fails. List all the nodes whose finger tables need to be updated.