5. **Randomized Algorithms**

Consider a situation where we have n servers and n clients. The servers all know a message m and the clients want to learn that message. Our goal is to design an algorithm that ensures that all clients learn m, while sending the smallest number of messages possible. We have access to a global random number generator R that generates a number uniformly at random between 1 and $\sqrt{n}$, and which all the servers can read.

Consider the following algorithm:

(a) Each client chooses a subset S of $k\sqrt{n}$ of the n servers, uniformly at random from all such subsets. The client then generates $k\sqrt{n}$ requests by choosing independently for each $s \in S$, a tag t that is an integer distributed uniformly at random between 1 and $\sqrt{n}$. The client sends each such request (s, t) to the server s

(b) A random number r is generated by the global random number generator and all servers read that number

(c) Every server s considers the requests they have received of the form (s, r). If there are less than $k\sqrt{n}$ such requests, then s sends m to each client that it received such a request from. k is a parameter to be determined later.

Unfortunately, some of the clients are *bad* in that they may disregard the first line of the algorithm, possibly sending out more than $k\sqrt{n}$ requests, and these requests may not necessarily be generated randomly. Note that the number of messages sent by each good client and server is always only $O(\sqrt{n})$. In this problem, you will show that even with the bad clients around, and even with this bound on communication costs, the protocol still has a good chance of ensuring all the good clients will learn m. (The algorithm thus has applications to mitigating situations like denial of service attacks by botnets.)

Call a sever *overloaded* if it receives $\geq k\sqrt{n}$ requests with tag r. Assume that each server receives at most n requests total, since if they receive ≥ 2 requests from the same client, they can throw out these requests, since that client is necessarily bad.

(a) (5 points) Derive an upper bound on the probability that a fixed server is overloaded.

(b) (5 points) Give an upperbound on the expected number of servers that are overloaded. Note: The events that two different servers are overloaded are NOT independent.

(c) (5 points) Now use Markov's inequality to bound the probability that the number of overloaded servers is greater than or equal to $n/6$. If everything is going well, you should be able to show this probability is no more than $6/k$.

(d) (5 points) Now assuming that at most $n/6$ servers are overloaded, calculate the probability that a given client fails to send a request to any server that is not overloaded. Note: The servers that a single client sends requests to are NOT chosen independently (since a given server can not be chosen more than once). You may find the following bound from the book helpful:

$$(x/y)^y \leq \binom{x}{y} \leq (xe/y)^y$$

Hint: In how many ways can you choose $k\sqrt{n}$ of the n servers? In how many ways can you choose $k\sqrt{n}$ of only the $n/6$ overloaded servers?

(e) (5 points) Finally, use union bounds and the above results to bound the probability that *any* good client does not receive the message. For n large, what is a good approximation to the probability of failure? Hint: Since the tags for good clients are chosen independently, you can use Chernoff bounds (see HW 2, problem 1) to bound the distribution of the number of requests from good clients that have tag r.