**Algorithm:**

K servers

N clients

w[i] for each client

distance\*w[i] is traffic for each client, sum of these traffics is total traffic.

While(true){

Loo0: judge whether server 0 should move, then move server 0 until it shouldn’t move.

Loop1: judge whether server 1 should move, then move server 1 until it shouldn’t move.

….

Loopk-1: judge whether server k-1 should move, then move server k-1 until it shouldn’t move.

if move happens, return to loop0 and go through each loop. Else, break.

}

**So, how to judge and move?**

We assume server[i] represent the location of server i. such that:

If sum( w[(server[i-1]+server[i])/2+1] +… + w[server[i]]) < sum( w[server[i]+1]+…+ w[(server[i]+server[i+1])/2]), server[i] ++

// that means the sum of left part of the server[i] is less than the sum of right part of the server[i].

Else: jump out of the loop.

The time complexity is O [KN],

prove: for each outer while loop, server 0 move first, then server 1, then server 2… finally server k-1;

Server 0 move more steps than other servers. So, for server 0, we assume for outer loop it first move step m0, second step m1, …finally step m, such that m0+m1+…+m <N. Such that for another server, the total number of comparing is less than N. So, the total number of comparing is less than KN.

So the time complexity is O[KN]

**Optimality:**

Assume we get an solution A,

IF there is another optimal solution B, such that there must exist a server which locate either too left or too right (or it will have the same location with the A). Whether it locate too right or too left, we could move this server to reduce the traffic, that means B is not an optimal solution, which contradict the optimality of B. So this solution is the only optimal solution.