TCP:  
aktív-e a szerver

TÁBLA  
szerver 1 | terheltség  
szerver 2 | terheltség  
…

2. IP és port elküldése

KLIENS 2

SZERVER 1

1. kapcso-lódás UDP-vel

KLIENS 4

KLIENS 3

KLIENS 1

SZERVER 2

NÉV SZERVER

ADATBÁZIS  
felhasználónév  
csatlakozva

- először a kliens csatlakozik UDP-vel a névszerverre

- a névszerver az, ami végig fut, és ő is indul el először (mint a chat alkalmazásnál a szerver)

- a névszerver kioszt neki egy IP-t és egy port-ot

- ez történhet körkörösen (egyszer szerver1-re irányítja, majd szerver2-re stb. S majd kezdi elölről)

- vagy irányíthatja a legkevésbé terhelt szerverre. Ebben az esetben a terheltséget is el kell tárolja a névszerver a táblában

- amikor a kliens csatlakozik a névszerverre

- elindít egy időzítőt, ha ez lejár és nem kap választ, akkor még kétszer megismétli (elküldi a kérést és elindítod az időzítőt).

- ha háromszor lejárt az időzítő, akkor nem elérhető a névszerver. Ezt kiírjuk.

- a duplikált csomagot nem kell lekezelni, mert nem érinti a feladatot

- ha megkapta az IP-t és a port-ot, csatlakozik az illető szerverre

- a szerver TCP-vel csatlakozik a névszerverre, mivel ha megszakad a kapcsolat, ez egyből észlelhető (vársz recieve-vel üzenetet, és ha megszakad, itt egyből észlelheted)

- hard code-olva (mint a chatnél), csatlakozik a szerver a névszerverre és majd utána a szerver elküld egy port-ot, amin keresztül a kliensek csatlakozhatnak rá

- amikor csatlakozott, bevezeti a névszerver a táblájába az illető szervert (itt most főleg a portot kell eltárolni, mivel az IP ugyanaz, mert egy gépről tesztelünk. Valós esetben az IP címek változnának, a port maradhatna ugyanaz)

- ha megszakad a kapcsolat, a névszerver tudja, hogy az illető szerver már nem aktív, és nem irányít oda klienseket (törli a táblából a szervert)

- a szerver elküldheti a terheltségi állapotát a névszervernek (hogy ha terheltség szerint szeretnénk kiosztani a klienseknek a szervereket)

- percenként

- vagy ha változott az állapota (csatlakozott rá egy kliens / lecsatlakoztak róla)

- a szerverek egy közös adatbázisban tárolják a kliensek felhasználóneveit

- ha csatlakozik a kliens, az adatbázisba bevezetjük az adatait

- ha csatlakozni szeretne, de már foglalt a név, innen tudhatja a szerver, és jelzi a kliensnek

- az adatbázis lehet sima mySQL, de lehet állomány is

- mindent osztályokban kell megcsinálni, különben legfeljebb 6-ost lehet kapni

- a kliensek csak egy szerveren keresztül kommunikálnak, azaz a kliens1 a kliens2-vel, és ők nem kommunikálnak a kliens3 és kliens4-gyel

- csak mindenkinek küldés kell legyen (nem kell privát üzenet stb.)

- bármilyen nyelvben íródhat az egész. Bármi bármilyen nyelvben lehet, de legyen objektumorientáltan

- leadási határidő: kb. 3 hét, de lehet vakáció után

- szálak

- az UDP-s részhez nem kell szál

-név szerver:

- egy TCP klienseket (ezek a szerverek) fogad, egy UDP klienseket (ezek a tényleges kliensek)

- tcp -> accept (ez az egyik szál)

- kliensnek egy szál

- és ahány szerver csatlakozik, annyi szál

- vagy lehet iteratikusan csinálni

- amit figyelembe vesz:

- hogy néz ki a kód

- mennyire strukturált

- mennyire tudod elmondani, hogy mit miért csináltál

- beszédes neveket használni

- hibakezelés

- kommunikációs protokoll