Industrijski komunikacioni protokoli u elektroenergetskim sistemima

Grupa A

U prilogu zadatka za praktični kolokvijum dati su izvorni kodovi **Client** i **Server** aplikacije koje komuniciraju putem TCP blokirajućih socketa.

Potrebno je razviti sistem u kojem merne stanice šalju svoja merenja servisu za prikupljanje podataka o indeksu kvaliteta vazduha. Servis ima mogućnost da komunicira sa najviše 10 mernih stanica i od njih prikuplja podatke. Periodično, servis ispisuje prosek indeksa kvaliteta vazduha za svaki grad posebno, i ukupan prosek za sve gradove.

Jedno merenje je u sistemu predstavljeno strukturom podataka **Measurement**.

```
#define BUFFER_SIZE 20
typedef struct Measurement_st {
        char city[BUFFER_SIZE];
        int index;
} Measurement;
```

Zadatak:

1. Nakon uspostave veze sa servisom, obezbediti unos i slanje merenja sa mernih stanica više puta ka servisu. Merne stanice rade u blokirajućem režimu. Pri tome, svaka merna stanica može poslati merenja za bilo koji grad, unosom grada i trenutnog indeksa kvaliteta vazduha za taj grad. Unosom ključne reči “Kraj”, merna stanica će pre zatvaranja sprečiti prijem i slanje podataka sa servisom slanjem komande -funkcija **shutdown(),** a potom se bezbedno zatvoriti.
2. Obezbediti neblokirajući režim izvršavanja operacija na servisu za prikupljanje podataka koristeći **metodu za multipleksirano praćenje događaja (select).** Pri tome:
    - Servis će istovremeno moći da prima merenja sa mernih stanica i prihvata nove zahteve za konekciju od mernih stanica.
    - Ukoliko je istovremeno konektovano **10** mernih stanica na servis, servis više neće prihvatati zahteve za konekciju sve dok bar jedna merna stanica ne raskine vezu. Dakle, **servis prima merenja od najviše 10 mernih stanica istovremeno**, uz mogućnost prihvatanja novih zahteva za konekciju ukoliko se smanji broj konektovanih mernih stanica od zadatog maksimuma.
    - Ako merna stanica pošalje komandu za onemogućavanje prijema i slanje podataka sa servisom **(shutdown)**, servis će bezbedno zatvoriti socket putem kojeg je komunicirao sa mernom stanicom, i normalno nastaviti sa prikupljanjem podataka sa ostalih mernih stanica.
3. Implementirati **thread-safe** strukturu podataka tipa **Queue**.
    - Nakon svakog prijema merenja sa bilo koje merne stanice, servis za prikupljanje podataka će smestiti merenje na red.
4. U posebnoj programskoj niti, servis za prikupljanje podataka meri vreme, od samog startovanja servisa. Na svakih **45 sekundi**, servis periodično skida sva merenja sa reda koja su do tada pristigla i računa prosek indeksa kvaliteta vazduha za svaki grad posebno, i ukupan prosek za sve gradove. Zatim, na konzolu ispisuje **timestamp** (datum i vreme), kao i proračunate proseke indeksa kvaliteta vazduha.

Prilog:

U prilogu je dat deo koda za ispis vremena.

```
#include <time.h>
```

Ispis vremena

```
time_t rawtime;
struct tm * timeinfo;
time ( &rawtime );
timeinfo = localtime ( &rawtime );
printf ( "Current local time and date: %s", asctime (timeinfo) );
```

Napomene:

- Voditi računa o rukovanju sa **HANDLE**-ovima i **SOCKET**-ima.
- Ukoliko se u toku komunikacije desi greška, bezbedno zatvoriti stranu na kojoj se greška desila.
- Obezbediti nedeljiv pristup svim deljenim resursima na serveru.
- Voditi računa o redosledu bajtova u komunikaciji **(host and network order)**.