Diplomski studij

Informacijska i komunikacijska tehnologija
Telekomunikacije i informatika

Računarstvo
Računarska znanost
Programsko inženjerstvo i informacijski sustavi

Ak. g. 2017./2018.

Aleksandar Antonić
Martina Marjanović
F. Matteo Benčić

Raspodijeljeni sustavi

Upute za izradu 1. domaće zadaće

Prikupljanje i obrada senzorskih podataka u Internetu stvari

Sadržaj

1 Uvod

CILJ DOMAĆE ZADAĆE:

U praksi utvrditi i ponoviti gradivo s predavanja. Studenti će naučiti programski izvesti raspodijeljeni sustav temeljen na modelu klijent-poslužitelj koristeći dvije vrste komunikacijske međuopreme (Socket API za protokol TCP i web-usluge). Za pohranu podataka koristit će se vlastita implementacija strukture podataka slična tehnologiji *Blockchain*.

ZADATAK:

Ova domaća zadaća se sastoji od sljedeća 3 dijela:

1. proučavanje primjera s predavanja (Socket API za TCP i web-usluge),
2. programiranje *poslužitelja raspodijeljenog sustava*,
3. programiranje *klijenta raspodijeljenog sustava*

Studenti trebaju programski izvesti klijenta i poslužitelja opisanog raspodijeljenog sustava.

PREDAJA:

Studenti su dužni u zadanom roku putem sustava *Moodle* predati arhivu koja se sastoji od sljedećih dijelova:

1. izvorni kod poslužitelja,
2. izvorni kod klijenta.

Navedene komponente trebaju biti realizirane u nekom od objektno-orijentiranih programskih jezika kao što su Java, C++, C# itd. s ogradom da su primjeri s predavanja izvedeni u programskom jeziku Java. Arhiva s izvornim kodom treba biti imenovana „Ime_Prezime“ (bez dijakritičkih znakova), a unutra se trebaju nalaziti 2 mape (jedna za klijenta, druga za poslužitelja) s datotekama izvornog koda. Napominjemo da datoteke s izvornim kodom trebaju biti smještene u odgovarajućoj mapi bez korištenja podmapa, tj. na kraju izrade domaće zadaće sve datoteke s izvornim kodom kopirajte u odgovarajuću mapu, npr. sudent Ivo Ivić predaje arhivu nazvanu „Ivo_Ivic.zip“ koja sadrži 2 mape („Klijent“ i „Posluzitelj“), a unutar svake mape nalaze se **SVE** datoteke s izvornim kodom, primjer strukture je:

/Klijent/Datoteka1.java

/Klijent/Datoteka2.java

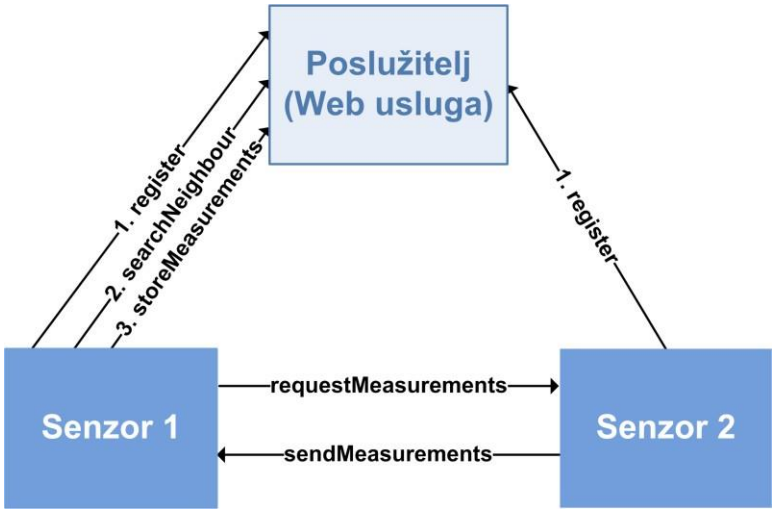
/Posluzitelj/Datoteka1.java

/Posluzitelj/Datoteka2.java

Osim predaje samih datoteka u digitalnom obliku, bit će organizirana i usmena predaja na kojoj će se ispitivati razumijevanje koncepata potrebnih za izradu domaće zadaće te poznavanje vlastitog programskog koda. Svi studenti trebaju proučiti primjere s predavanja, a moguće je da pri usmenoj predaji bude ispitivano i znanje studenta o tim primjerima.

Studenti koji budu kasnili s predajom će dobiti 0 bodova iz domaće zadaće. Studenti koji budu mijenjali programski kod nakon predaje na Moodle dobit će 0 bodova na usmenom kolokviranju zadaće. Za sve studente su organizirane konzultacije za izradu programskog rješenja petkom od 11:00 do 12:00 u laboratoriju IoT-lab (C08-18).

2 Arhitektura raspodijeljenog sustava



Slika 1 - Arhitektura sustava

Raspodijeljeni sustav služi za praćenje senzorskih očitanja na nekom zemljopisnom području uz pomoć poslužitelja koji će biti implementiran kao web-usluga i skupine klijenata na koje pretpostavljamo da su spojeni senzori. *Klijent raspodijeljenog sustava* (od sada nadalje se koristi riječ ***senzor***) se sastoji od klijenta web-usluge, TCP klijenta i TCP poslužitelja. Cilj senzora je da svoja očitanja usporede s očitanjima geografski najbližih senzora jer su senzori često podložni ispadima, te da svoja „umjerena“ očitanja prijave *poslužitelju web-usluge* (od sada nadalje se koristi riječ ***poslužitelj***). Dakle, sustav se sastoji od jednog poslužitelja i više instanci senzora (prave senzore nećemo spajati u sustav, već će se emulirati generiranje podataka na temelju pripremljenih očitanja iz datoteke). Uloga poslužitelja je da održava podatke o registriranim raspoloživim senzorima, njihovim geografskim lokacijama i prijavljenim umjerenim očitanjima u strukturi podataka sličnoj onoj koju koristi tehnologija *Blockchain* (svi podaci se održavaju u radnoj memoriji poslužiteljskog računala). Osim komunikacije s poslužiteljem, senzori mogu komunicirati i međusobno koristeći protokol TCP.

Senzor se prilikom pokretanja registrira kod poslužitelja, a nakon toga pozivom odgovarajuće metode poslužitelja može pronaći geografski najbliži senzor za umjeravanje. Senzor traži najbližeg susjeda kako bi provjerio kvalitetu vlastitog mjerenja uspoređujući ga s mjerenjem susjednog senzora. Zatim umjereno očitanje koje računa kao prosječnu vrijednost vlastitog i primljenog očitanja od najbližeg susjeda šalje poslužitelju koristeći odgovarajuću metodu web-usluge.

Napomena: Prilikom ostvarivanja ove dvije vrste komunikacije posebnu pozornost obratite na to da senzor i poslužitelj moraju biti realizirani u dva odvojena projekta odabrane razvojne okoline (npr. Eclipse, NetBeans, Visual Studio), ali zato projekti trebaju dijeliti zajednička sučelja (sučelje web-usluge).

3 Komunikacija između senzora i poslužitelja koristeći web- usluge

Po uzoru na prikazani postupak s predavanja izvedite komunikaciju senzora i poslužitelja koji komuniciraju koristeći web-usluge. Poslužitelj treba implementirati 3 metode. Jedna metoda će služiti za registraciju senzora kod poslužitelja (**`boolean register(String username, double latitude, double longitude, String IPaddress, int port)`**), druga metoda vraća informacije o geografski najbližem senzoru među senzorima koji su trenutno spojeni na poslužitelj (**`UserAddress search Neighbour(String username)`**), a treća metoda služi za prijavljivanje umjerenih podataka (**`boolean storeMeasurement(String username, String parameter, float averageValue)`**). Neka se prilikom prijavljivanja umjerenih podataka ažurira i stanje blok-lanca. U blok se osim prijavljenih umjerenih podataka zapisuje i vremenski trenutak kada je poslužitelj primio umjerene podatke, što definira i slijed zapisa u strukturi blok-lanca.

Udaljenost između dva senzora odredite korištenjem Haversinove formule koja je definirana sljedećim pseudo kodom (6371 km je Zemljin radijus):

```
R = 6371
dlon = lon2 - lon1
dlat = lat2 - lat1
a = (sin(dlat/2))^2 + cos(lat1) * cos(lat2) * (sin(dlon/2))^2
c = 2 * atan2( sqrt(a), sqrt(1-a) )
d = R * c
```

4 TCP komunikacija između dva senzora

Prilikom izrade domaće zadaće potrebno je maksimalno iskoristiti programski kôd s predavanja te programski izvesti dohvaćanje mjerenja od susjeda koristeći Socket API za komunikacijski protokol TCP. Senzori moraju biti u mogućnosti održavati više konkurentnih konekcija, tj. istodobno komunicirati s više senzora koji od njih traže očitanja te po potrebi otvoriti novu konekciju prema susjednom senzoru.

Mjerenje senzora će se emulirati slučajnim dohvaćanjem vrijednosti iz priložene ulazne datoteke (`mjerenja.csv`). Ulazna datoteka je strukturirana datoteka koja sadrži informacije o temperaturi, tlaku zraka, relativnoj vlažnosti te plinovima CO, NO_2 i SO_2. Senzori koji su korišteni za stvaranje ulazne datoteke sadrže samo dva senzora plina (CO, NO_2 ili SO_2), te prilikom učitavanja ulazne datoteke vodite računa da mogu nedostajati vrijednost za pojedini plin.

Nakon pokretanja, senzor učitava ulaznu datoteku i na slučajan način odabire jednu geografsku lokaciju iz intervala [15.87, 16] za Longitude i [45.75, 45.85] za Latitude (ova lokacija se ne mijenja tijekom životnog ciklusa klijenta), a svoj identifikator definira proizvoljno. S tim podacima senzor se registrira kod poslužitelja i čeka daljnje naredbe (bilo od krajnjeg korisnika ili od drugih senzora) koje će kontinuirano pratiti do zaustavljanja.

1. Krajnji korisnik može započeti proces mjerenja, a to u praksi znači da senzor na slučajan način prvo generira očitanje iz datoteke (postupak je definiran u nastavku), potom otvara konekciju prema susjedu kako bi dohvatio njegova očitanja te prijavio umjerena očitanja poslužitelju. Potom se cijeli postupak na senzoru ponavlja (generiranje očitanja, dohvat podataka od susjeda i slanje podataka na poslužitelj) do njegovog zaustavljanja od stane krajnjeg korisnika.
2. Senzor treba moći primiti zahtjev za otvaranjem konekcije od svojih susjeda-senzora te kontinuirano slati očitanja generirana iz datoteke do zahtjeva za zaustavljanjem kojega definira njegov susjed ili do naprasnog prekida konekcije. Kada dođe do prekida TCP konekcije na sučelju senzora treba ispisati odgovarajuću poruku.

Proces mjerenja senzor emulira dohvaćajući podatke iz priložene datoteke (`mjerenja.csv`). Za generiranje očitanja iz ulazne datoteke koristite sljedeću formulu kako biste dobili redni broj retka datoteke iz kojega ćete očitati mjerenja za temperaturu, tlak zraka, relativnu vlažnost te plinove CO i NO_2 ili SO_2:

```
Int redni_broj = (broj_aktivnih_sekundi % 100) + 2
```

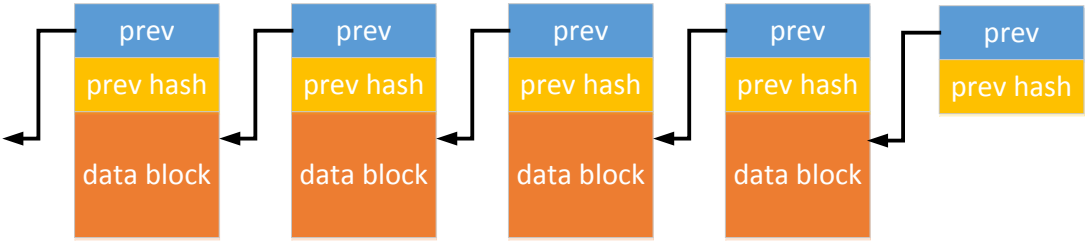
Broj aktivnih sekundi je vrijeme koje je senzor aktivan (to vrijeme se računa od pokretanja samog klijenta i izražava u sekundama). Koristite operaciju ostatak (*mod)* 100 i dobivenu vrijednost uvećajte za 2. Dobiveni broj predstavlja redak ulazne datoteke iz kojeg treba dohvatiti mjerenja. Ovaj postupak emuliranja mjerenja treba koristiti u oba slučaja: kada senzor želi generirati vlastito očitanje i za postupak kada je potrebno očitati mjerenje na zahtjev drugog senzora. Prilikom mjerenja potrebno je na sučelju senzora ispisati sve informacije (*broj_aktivnih_sekundi* i izračunati *redni_broj* korištenog retka datoteke) te dohvaćene podatke iz datoteke.

Obratite pozornost da je poslužitelju potrebno dostaviti samo prethodno umjerene podatke, a *umjeravanje* izvršite računanjem prosječne vrijednosti između dva mjerenja (jedno je očitanje samog senzora, a drugo njegovog susjeda). Nakon dohvaćanja vlastite izmjerene vrijednosti, drugo mjerenje, koje služi za umjeravanje, dohvatite (koristeći TCP konekciju) od geografski najbližeg susjeda (informaciju o adresi i vratima na kojima susjed sluša dolazne zahtjeve dobivate od poslužitelja). Dohvaćanje mjerenja od susjeda započinje slanjem zahtjeva susjedu (sami definirajte formu zahtjeva), a susjedni senzor će odgovoriti slanjem vlastitog mjerenja. Za dohvaćanje uzastopnih mjerenja koristite istu TCP konekciju (konekcija se zatvara tek prilikom zaustavljanja procesa mjerenja).

Obratite pažnju da prilikom umjeravanja računate prosječnu vrijednost za isti tip mjerenja (za temperaturu, tlak zraka, relativnu vlažnost te plinove CO, NO_2 i SO_2), a ako nedostaju vrijednosti za umjeravanje (slučaj sa susjednim senzorima koji npr. mjere SO_2 i NO_2 pa nije moguće izračunati srednju vrijednost) odnosno ako je izmjerena vrijednost 0 (tj. postoji greška u očitanju parametra na senzoru), tada objavite samo vlastitu vrijednost mjerenja.

5 Pohranjivanje podataka na poslužitelju u *Blockchain*

Struktura blok-lanca je slična jednostruko povezanoj listi. Načinjen je od niza *blokova* od kojih svaki osim prvog sadrži sažetak (*hash*) prethodnog bloka i na taj način sprječava falsifikaciju blokova. U podatkovni blok (*data block*) pohranjuju se proizvoljne informacije.



Slika 2 - Struktura blok-lanca

Uobičajeno, svaki je podatkovni blok sačinjen od više informacija (u kontekstu blok-lanca nazivaju se *transakcija*). Transakcija može sadržavati proizvoljne podatke, a u kontekstu zadaće jedna transakcija predstavlja jedan prijavljeni umjereni podatak i vremenski trenutak kada je poslužitelj primio podatak. Pojednostavljenja radi, neka jedan podatkovni blok sadrži samo jednu transakciju, tj. samo jedno prijavljeno mjerenje.

Naša će implementacija biti centralizirana, što znači da centralni autoritet održava *Blockchain*. Centralizacija za sobom povlači dva negativna svojstva: centralni autoritet mogao bi modificirati podatke primljene od klijenata prije pohrane ili bi ih mogao u potpunosti odbaciti odnosno selektivno pohranjivati. Problem modifikacije rješava se digitalnim potpisivanjem. Problem selekcije rješava se raspodijeljenom implementacijom Blockchaina – u zadaći pretpostavljamo da vjerujemo centralnom autoritetu da neće selektivno pohranjivati podatke niti će ih modificirati, pa ne koristimo niti digitalni potpis niti raspodijeljenu implementaciju blok-lanca.

Dodavanje novog mjerenja u blok, a zatim u Blockchain, vrši se metodom **`boolean append(String username, String parameter, float averageValue)`** Klasa **`Block`** sadrži sažetak prethodnog bloka i podatkovni blok. Blokove je dovoljno pohraniti u listu. Metoda **`Block peekLast()`** vraća zadnji blok dok metoda **`Block getBlock(int i)`** vraća blok pod rednim brojem i. Klasa **`Measurement`** sadrži sva mjerenja koja je prijavio jedan senzor u nekom trenutku. Metoda **`Set<Measurement> getState()`** vraća kolekciju posljednjih mjerenja **svih registriranih senzora** tj. vraća vrijednosti koje su posljednje zabilježene u blok-lanac za sve fizikalne veličine koje senzor mjeri i to za sve registrirane senzore (npr. *Senzor1* mjeri *CO=1* i temperatura=20, a *Senzor2* mjeri *CO=0.7,* temperatura=23 i vlaga=75). Taj skup posljednjih očitanja možemo smatrati *trenutnim globalnim stanjem sustava*. Globalno stanje sustava rekonstruirajte prateći transakcije zabilježene u Blockchainu.