Diplomski studij

Informacijska i komunikacijska tehnologija

Telekomunikacije i informatika

Računarstvo

Programsko inženjerstvo i informacijski sustavi

Računarska znanost

Ak. g. 2020./2021.

Raspodijeljeni sustavi

Upute za izradu 2. domaće zadaće
Praćenje senzorskih očitanja u vremenu

Sadržaj

1. Uvod

CILJ DOMAĆE ZADAĆE

U praksi utvrditi i ponoviti gradivo s predavanja. Studenti će naučiti programski izvesti decentralizirani raspodijeljeni sustav s ravnopravnim sudionicima koristeći komunikaciju protokolom UDP. Pritom će primijeniti mehanizam za sinkronizaciju procesa tijekom vremena.

ZADATAK

Ova domaća zadaća sastoji se od sljedeća 2 dijela:

1. proučavanje primjera s predavanja (komunikacija protokolom UDP),
2. programiranje čvora koji je dio sustava s ravnopravnim sudionicima.

Studenti trebaju programski izvesti čvor opisanog raspodijeljenog sustava, a prilikom demonstracije rješenja će pokrenuti više instanci čvora (minimalno 3 procesa, ali mora biti moguće pokrenuti i više procesa).

PREDAJA

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

1. izvornog kod čvora sustava s ravnopravnim sudionicima.

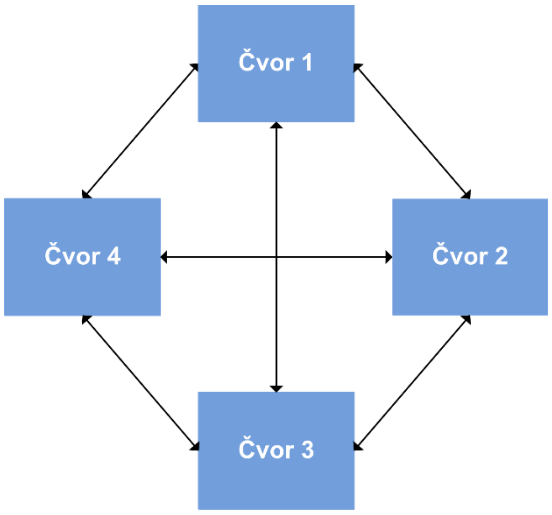
Navedene komponente trebaju biti realizirane u nekom od objektno-orijentiranih programskih jezika kao što su Java, C++, C# itd. s ogradom da su primjeri s predavanja izvedeni u programskom jeziku Java. Očekuje se poštivanje dobre programerske prakse. Arhiva s izvornim kodom treba biti imenovana „Ime_Prezime“ (bez dijakritičkih znakova), a unutra se treba nalaziti mapa s datotekama izvornog koda.

Dozvoljeno je predati .zip exporta projekta u slučaju da je zadaća implementirana u IDE-u (Eclipse, NetBeans, itd.).

Osim predaje datoteka u digitalnom obliku, bit će organizirana i usmena predaja putem sustava Microsoft Teams 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, odnosno koji će prilikom usmenog odgovaranja koristiti izvorni kod neistovjetan predanom kodu na Moodleu, dobit će 0 bodova iz domaće zadaće. Za sve studente su organizirane konzultacije za izradu programskog rješenja utorkom od 10:00 do 11:00 putem sustava Microsoft Teams, uz najavu e-mailom.

2. Arhitektura raspodijeljenog sustava



Raspodijeljeni sustav služi za praćenje senzorskih očitanja u vremenu uz pomoć skupine ravnopravnih čvorova na koje pretpostavljamo da su spojeni senzori. Za usporedbu možete ovo programsko rješenje usporediti s bežičnim mrežama senzora (npr. Waspmote ili Arduino moduli) koji međusobno razmjenjuju očitanja, a **trebaju ih sortirati u vremenu i izračunati srednje vrijednosti očitanja u prozoru od 5 sekundi**. *Čvor raspodijeljenog sustava* (od sada nadalje koristi se riječ **čvor**) sastoji se od **UDP klijenta** i **UDP poslužitelja**. Cilj čvorova je da međusobno razmjenjuju očitanja i nakon nekog vremenskog perioda ih **sortiraju u vremenu koristeći skalarne i vektorske oznake vremena**. Za taj vremenski period je potrebno **izračunati srednje vrijednosti očitanja**. Cilj je da svi čvorovi sortiraju primljena i vlastita očitanja na istovjetan način, a da pritom to odgovara stvarnom redoslijedu događaja primanja i slanja poruka među čvorovima.

Kako biste omogućili da se čvorovi povežu u mrežu čvorova, prilikom pokretanja definirajte statičnu konfiguraciju koja će definirati sve čvorove sustava (pretpostavka je da je riječ o dobro povezanoj mreži, tj. svi čvorovi su međusobno povezani). To možete ostvariti tako da npr. nakon pokretanja čvor učita konfiguracijsku datoteku u kojoj su navedeni podaci o ostalim čvorovima ili da korisnik ručno unosi podatke o svim ostalim čvorovima. Nakon ostvarivanja mreže čvorova, čvorovi mogu pokrenuti proces očitavanja podataka.

Dakle, sustav se sastoji od **više instanci ravnopravnih čvorova** na koje su spojeni senzori (prave senzore nećemo spajati u sustav, već će se emulirati generiranje podataka na temelju pripremljenih očitanja iz datoteke) koji međusobno komuniciraju koristeći **protokol UDP**.

Napomena: Posebnu pozornost obratite na to da je čvorove potrebno realizirati u zasebnim procesima kako bi se omogućilo njihovo pokretanje na 2 ili više računala, tj. **NIJE** dozvoljeno pokretanje više čvorova unutar jednog procesa.

3. UDP komunikacija između čvorova

Prilikom izrade domaće zadaće studenti trebaju maksimalno iskoristiti programski kod s predavanja te programski izvesti komunikaciju protokolom UDP. Protokol UDP ne osigurava pouzdanu komunikaciju od točke do točke, već je njena realizacija prepuštena višim slojevima (tj. aplikacijskom sloju).

Kako će se komunicirajući čvorovi izvoditi na istom računalu te između njih neće biti stvarne mreže, potrebno je na neki način simulirati stvarnu mrežu u kojoj se paketi mogu izgubiti i stići različitim redoslijedom od redoslijeda slanja. Za simuliranje komunikacije stvarnom mrežom **potrebno je koristiti priloženu klasu `SimpleSimulatedDatagramSocket`** koju možete pronaći u arhivi koja je dostupna na stranici predmeta u mapi *Domaće zadaće* (*Domaće zadaće* > *2. DZ* > Programski primjeri uz domaću zadaću – UDP > *StupidUDPClient/StupidUDPServer*). Ova klasa koristi se na način identičan onome klase `DatagramSocket`, koji je bio objašnjen na predavanju. Razlika između ove dvije klase je u tome što klasa `SimpleSimulatedDatagramSocket` ima malo drugačiji konstruktor: `SimpleSimulatedDatagramSocket(double lossRate, int averageDelay)`. Ovaj konstruktor prima sljedeća 2 parametra: **postotak izgubljenih paketa u mreži [postotak] i prosječno kašnjenje paketa u jednom smjeru [milisekunde]**. Parametar kašnjenja paketa postavite na **1000 ms**, a postotak izgubljenih paketa u mreži na **0.3 (30%)** ili više ako želite češću retransmisiju.

Za emuliranje različitih fizičkih satova na istom računalu potrebno je koristiti priloženu klasu `EmulatedSystemClock`. Instancu klase kreirajte prilikom pokretanja čvora te ju koristite za dohvaćanje skalarne oznake vremena (metoda `currentTimeMillis()`).

Napomena: Ukoliko se student odluči za neki drugi programski jezik osim Jave, tada će morati simulirati stvarnu mrežu po uzoru na način ostvaren klasama `SimpleSimulatedDatagramSocket` i `EmulatedSystemClock`, koju možete modificirati ili dekorirati po potrebi.

Čvor očitava senzorska očitanja svake sekunde, pakira ih u pakete kojima prilikom slanja pridjeljuje ažurnu vektorsku i skalarnu oznaku vremena i šalje ih svim čvorovima u mreži. U paraleli s očitavanjem senzorskih očitanja, čvor čeka potvrde za sve poslane pakete te po potrebi (u slučaju ne primanja potvrde) ponovno šalje paket za koji nije primio potvrdu. Za svako uspješno primljeno očitanje, Čvor X će Čvoru Y poslati potvrdu (također u obliku paketa, podrazumijeva se da je potvrda jednostavnijeg formata od očitanja). Čvor Y treba voditi brigu o izgubljenim paketima te ponoviti slanje svih izgubljenih paketa. **Prilikom ponovnog slanja paketa, potrebno je zadržati originalne oznake vremena**. Moguća je i situacija u kojoj će poneki paket biti poslan (i primljen) nekoliko puta te tada takve pakete treba zanemariti, ali treba poslati potvrdu da je paket primljen. **Čvor za primanje potvrda i dolaznih očitanja susjeda koristi ista vrata odnosno potrebno je na aplikacijskoj razini implementirati logiku koja prepoznaje radi li se o potvrdi ili o očitanju te ispravno obraditi takav paket**.

Klasu `SimpleSimulatedDatagramSocket` treba koristiti prilikom slanja paketa s obje strane te je stoga moguće i da neke potvrde budu izgubljene.

Svaki čvor treba voditi brigu o redoslijedu primljenih očitanja i sva dolazna očitanja pohranjivati u radnoj memoriji. Nakon 5 sekundi **svaki čvor sortira sva očitanja** (vlastita i primljena od drugih čvorova u mreži) **koristeći vektorske i skalarne oznake vremena** te ih **ispisuje na ekran**. Također **računa i ispisuje srednju vrijednost primljenih očitanja tijekom prozora od 5 sekundi**. **Sortiranje je potrebno provesti koristeći obje oznake vremena** te **pojedinačno ispisati svaki sortirani skup**. Nije potrebno sortirati sva očitanja od pokretanja čvorova već se sortiraju samo očitanja primljena u zadnjih 5 sekundi.

Mjerenje senzora će se emulirati dohvaćanjem vrijednosti samo za ugljični monoksid (CO) iz ulazne datoteke (`mjerenja.csv`) koja je dostupna na stranici predmeta (*Domaće zadaće* > *2. DZ* > *mjerenja.csv*). Ulazna datoteka je strukturirana datoteka koja sadrži informacije o temperaturi, tlaku zraka, relativnoj vlažnosti te plinovima CO, NO_2 i SO_2.

Proces očitanja čvor emulira dohvaćajući podatke svake sekunde iz priložene datoteke. Za generiranje očitanja iz ulazne datoteke koristite sljedeću formulu kako biste dobili redni broj retka datoteke iz kojega ćete očitati mjerenja za CO:

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

Broj aktivnih sekundi je vrijeme koje je čvor aktivan (to vrijeme se računa od pokretanja čvora 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.