FER3

Diplomski studij

Raspodijeljeni sustavi

Druga laboratorijska vježba — Praćenje senzorskih očitanja u vremenu

Ak. god. 2021./2022.

1 Uvod

Cilj ove vježbe je izgraditi decentralizirani raspodijeljeni sustav s ravnopravnim sudionicima (od sad nadalje se koristi riječ **čvor**) koristeći komunikaciju protokolom UDP i Kafka[1] kontrolnim čvorom kako je definirano u odjeljku 2, te naučiti praktičnu uporabu tehnologije Kafka. Pritom će se primjeniti mehanizam za sinkronizaciju procesa tijekom vremena. Ova se vježba sastoji od tri dijela:

- Proučavanja priloženih primjera koda s predavanja (Kafka i komunikacija protokolom UDP).
- Programiranje čvora koji je dio sustava s ravnopravnim sudionicima.
- Programiranje Kafka kontrolnog čvora.

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

1.1 Predaja

Studenti su dužni putem sustava Moodle predati arhivu koja se sastoji od:

- Izvornog kôda čvora sustava s ravnopravnim sudionicima.
- Izvornog kôda Kafka upravljačkog čvora.

Arhiva se mora predati do roka koji će biti objavljen na stranici predmeta. Uz predaju digitalne arhive, organizirat će se usmena prezentacija vaše predaje, gdje će studenti morati objasniti implementaciju i pokazati svoje razumijevanje primijenjenih komunikacijskih načela. Termin usmenog ispita bit će objavljen na stranici predmeta.

Sve komponente sustava mogu se implementirati u bilo kojem programskom jeziku. Imajte na umu da su primjeri prikazani na predavanjima implementirani u programskom jeziku Java. Arhivu, koja sadrži izvorni kôd, treba nazvati **ime_prezime_jmbag**. Arhiva treba sadržavati 1 direktorij koji sadrži datoteke izvornog kôda. Dopušteno je slanje zip arhive koja sadrži izvoz projekta ako je implementiran u IDE-u (Eclipse, NetBeans itd.).

Studenti koji predaju vježbu nakon navedenog roka dobit će 0 bodova iz vježbe. Konzultacije se održavaju **utorkom od 10:00 do 11:00**, uz prethodnu **najavu** na MS Teamsu (kanal Laboratorijske vježbe) ili e-mailom.

2 Arhitektura raspodijeljenog sustava

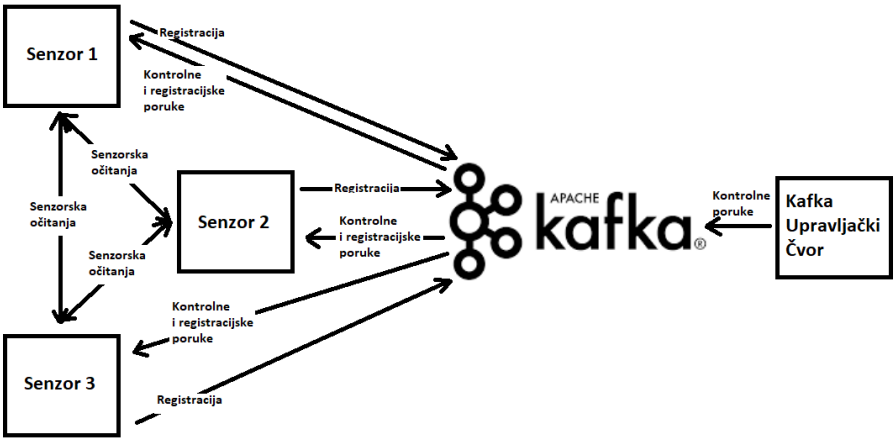
Kratak pregled funkcionalnosti sustava: Raspodijeljeni sustav koristi se za praćenje senzorskih očitanja u vremenu uz pomoć skupine ravnopravnih čvorova, gdje svaki čvor **"glumi"** senzora. Čvorovima se upravlja pomoću kontrolnih poruka koje šalje Kafka upravljački čvor (odnosno **koordinator**). Pregled arhitekture raspodijeljenog sustava prikazan je na slici 1.

Zadaća koordinatora i poslužitelja Kafka je kreirati potpuno povezanu mrežu čvorova, tj. omogućiti **razmjenu identifikatora** čvorova (kako bi se čvorovi mogli povezati) te **pokretanje** i **zaustavljanje** čvorova. Funkcionalnost koordinatora i komunikacija koordinatora i čvorova detaljno su objašnjeni u poglavlju 4.

Zadaća čvorova je razmjena senzorskih očitanja te **sortiranje primljenih i vlastitih očitanja** u vremenu koristeći **skalarne i vektorske** oznake vremena te **izračunavanje srednje vrijednosti** očitanja u vremenskom prozoru od **5 sekundi**. Senzori međusobno komuniciraju protokolom **UDP**,

[1] https://kafka.apache.org/

odnosno svaki senzor je **UDP klijent** i **UDP poslužitelj**. Dodatno, svaka UDP konekcija simulira gubitak UDP paketa u mreži. Svaki izgubljeni UDP paket potrebno je ponovno poslati, odnosno potrebno je implementirati **retransimiju**. Funkcionalnost čvorova i komunikacija između čvorova detaljno su objašnjeni u poglavlju 5.



Slika 1: Arhitektura raspodijeljenog sustava

Napomena: Čvorove je 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 Instalacijske upute

U nastavku se opisuje instalacija Kafke, Zookepera (alata praćenje statusa čvorova spojenih na Kafku te za održavanje tema i poruka) te postavljanje razvojne okoline za operacijski sustav Windows. Preduvijet za pokretanje Zookeeper i Kafka poslužitelja je **Java 8** SDK ili novija verzija. Najnovije verzije Zookeeper i Kafke se mogu preuzeti na **poveznici**. Arhivu ekstraktajte u proizvoljni direktorij (u daljnjem tekstu **KAFKA_HOME**). Primjer vrijednosti **KAFKA_HOME** varijable je **`C:\kafka`**.

Napomena: Preporučljivo je da koristite verziju Kafke 2.8.1 Scala 2.13, a ne 3.0.0 jer ta verzija nema dobru podršku za Windows datotečni sustav.

3.1 Kafka

Kafka se konfigurira korištenjem konfiguracijske datoteke **server.properties** koja se nalazi u **`KAFKA_HOME\config`** direktoriju (npr. **`C:\kafka\config\server.properties`**). Kafka pohranjuje sadržaj na lokaciju koja je definirana u datoteci server.properties pod vrijedošću **log.dirs**. Vrijednost **log.dirs** postavite proizvoljno, npr. **`KAFKA_HOME\data\kafka`**. Sve ostale postavke mogu ostati na pretpostavljenim vrijednostima.

3.2 Zookeeper

Zookeeper se konfigurira se korištenjem konfiguracijske datoteke **zookeeper.properties** koja se nalazi u **`KAFKA_HOME\config`** direktoriju (npr. **`C:\kafka\config\zookeeper.properties`**. Zookeeper pohranjuje sadržaj na lokaciju koja je definirana u datoteci zookeeper.properties pod vrijedošću **dataDir**. Vrijednost **dataDir** postavite proizvoljno, npr. **`KAFKA_HOME\data\zookeeper`**. Sve ostale postavke mogu ostati na pretpostavljenim vrijednostima.

3.3 Pokretanje Zookeeper i Kafke poslužitelja

Zookeeper i Kafka pokreću se korištenjem terminala. Prvo je potrebno pokrenuti Zookeeper, a nakon njega Kafku. Skripte za pokretanje nalaze se u **`KAFKA_HOME\bin\windows`** direktoriju. Zookeeper se pokreće korištenjem **zookeeper-server-start** skripte koja kao argument prima putanju do konfiguracijske datoteke (npr. **`C:\kafka\bin\windows\zookeeper-server-start.bat C:\kafka\config\zookeeper.properties`**). Kafka se pokreće korištenjem **kafka-server-start** skripte koja kao argument prima putanju do konfiguracijske datoteke (npr. **`C:\kafka\bin\windows\kafka-server-start.bat C:\kafka\config\server.properties`**).

4 Funkcionalnost koordinatora

Ključne funkcije Kafka upravljačkog čvora jesu slanje kontrolnih poruka za razmjenu identifikatora, pokretanje i zaustavljanje grupe čvorova. Radi jednostavnosti, možete pretpostaviti da su prije slanja prve poruke svi čvorovi pretplaćeni na odgovarajuće teme, te da nakon pokretanja čvorova u sustav neće biti dodani novi čvorovi.

Napomena: Kafka poslužitelj i koordinator su dva različita procesa. Koordinator je objavitelj poruka, dok Kafka poslužitelj prima te poruke i prosljeđuje ih dalje.

4.1 Pokretanje grupe čvorova

Koordinator pokreće grupu čvorova slanjem kontrolne poruke **"Start"** koja se šalje na temu **"Command"**. Funkcionalnost čvorova prije i nakon primanja poruke **"Start"** objašnjeno je u poglavljima 5.1 i 5.2.

4.2 Zaustavljanje grupe čvorova

Koordinator zaustavlja grupu čvorova slanjem kontrolne poruke **"Stop"** koja se šalje na temu **"Command"**. Nakon slanja poruke **"Stop"**, koordinator se mora ugasiti. Funkcionalnost čvorova nakon primanja poruke **"Stop"** objašnjena je u poglavlju 5.8.

5 Funkcionalnost čvorova

Ključne funkcionalnosti čvora uključuju međusobno razmjenjivanje registracijske poruke, generiranje vlastitih očitanja, razmjenu vlastitih očitanja sa drugim čvorovima putem UDP veze, mogućnost sortiranja primljenih i vlastitih očitanja koristeći skalarne i vektorske oznake vremena i izračunavanje srednje vrijednosti očitanja u vremenskom prozoru od 5 sekundi. Prilikom implementacije čvora, možete koristiti priloženi kôd dostupan na stranicama predmeta kao primjer za razvoj UDP klijenta i UDP poslužitelja. Nakon što se formira povezana mreža čvorova, postupak generiranja očitanja, dohvaćanja očitanja od drugih čvorova, sortiranje i izračunavanje srednje vrijednosti se neprestano ponavlja sve dok koodrinator ne pošalje kontrolnu poruku za zaustavljanje grupe čvorova.

5.1 Inicijalizacija čvora

Nakon pokretanja, čvor traži upis **identifikatora** (odnosno **id**) i **porta** od korisnika. Nakon toga pretplaćuje se na teme **"Register"** i **"Command"** i čeka kontrolnu poruku **"Start"** od koordinatora. Radi jednostavnosti, pretpostavite da koordinator šalje poruku **"Start"** samo jedanput, te da nakon slanja poruke **"Start"** novi senzori neće biti dodani u sustav.

5.2 Registracija čvora

Nakon što čvor primi **"Start"** kontrolnu poruku, šalje registracijsku poruku na temu **"Register"** u JSON formatu. Registracijska poruka sastoji se od **identifikatora** čvora, njegove **IP adrese** i **porta**. Primjer registracijske poruke prikazuje JSON 1.

JSON 1: Primjer registracijske poruke u JSON formatu

```
{
    "id": "1",
    "address": "localhost",
    "port": "8000"
}
```

5.3 Uspostavljanje UDP komunikacije

Nakon što čvor primi registracijske poruke koje sadrže identifikatore svih ostalih čvorova putem teme **"Register"**, čvor započinje komunikaciju sa svim ostalim čvorovima korištenjem protokola UDP.

Za simuliranje gubitka paketa u mreži potrebno je koristiti klasu **SimpleSimulatedDatagramSocket** koju možete pronaći u arhivi na stranicama predmeta. Ova klasa koristi se na način identičan klasi DatagramSocket. Razlika između ove dvije klase je u tome što klasa **SimpleSimulatedDatagramSocket** ima drugačiji konstruktor: **SimpleSimulatedDatagramSocket(double lossRate, int averageDelay)**.

Ovaj konstruktor prima 2 parametra: postotak izgubljenih paketa u mreži te prosječno kašnjenje paketa u jednome smjeru izraženo u milisekundama. Postotak izgubljenih paketa u mreži na 30%, Parametar kašnjenja paketa postavite na $1000ms$.

Napomena: S obzirom na to da je sve komponente sustava dozvoljeno implementirati u bilo kojem programskom jeziku, funkcionalnost klase **SimpleSimulatedDatagramSocket** potrebno je reimplementirati u odabranom jeziku ako se ne koristi Java.

5.4 Generiranje senzorskih očitanja

Budući da pravi senzori nisu spojeni na sustav, njihova se funkcija emulira pripremljenim očitanjima iz ulazne datoteke **readings.csv**, gdje očitanja (retci) imaju više vrijednosti (stupaca). Za generiranje očitanja iz ulazne datoteke upotrijebite jednadžbu 1 za dobivanje broja retka koji ćete koristiti za očitavanje koncentracije plina CO iz ulazne datoteke:

$$red \leftarrow (brojAktivnihSekundi \ \% \ 100) + 1 \tag{1}$$

Varijabla *brojAktivnihSekundi* je vrijeme rada senzora (vrijeme se mjeri od trenutka pokretanja klijenta i izražava se u sekundama).

5.5 Razmjena očitanja i ažuriranje vremenskih oznaka

Nakon što čvor generira vlastito čitanje, stvara UDP paket koji sadrži generirano očitanje zajedno s **ažuriranom vektorskom i ažuriranom skalarnom vremenskom oznakom** (u daljnjem tekstu podatkovni paket), te ga šalje preko UDP-a svim čvorovima u mreži.

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

Napomena: S obzirom na to da je sve komponente sustava dozvoljeno implementirati u bilo kojem programskom jeziku, funkcionalnost klase **EmulatedSystemClock** potrebno je reimplementirati u odabranom jeziku ako se ne koristi Java.

Za svaki uspješno primljeni podatkovni paket, čvor X šalje potvrdu čvoru Y, također u obliku paketa (u daljnjem tekstu potvrda). Čvor Y čeka potvrde za sve pakete koje šalje.

Napomena: Potvrda mora biti jednostavnijeg formata od podatkovnog paketa. Svaki čvor mora koristiti **ista vrata** (eng. *port*) za primanje potvrda i podatkovnih paketa, tj. logika razlučivanja radi li se o potvrdi ili podatkovnom paketu mora biti implementirana **na aplikacijskoj razini** (npr. **nije dozvoljeno** koristiti jedna vrata za podatkovne pakete a druga za potvrde).

5.6 Retransmisija izgubljenih paketa

Ako čvor ne primi potvrdu za poslani podatkovni paket, ponovno šalje taj podatkovni paket. Prilikom ponovnog prijenosa podatkovnog paketa, izvorne vremenske oznake moraju biti sačuvane. Također je moguće da se neki podatkovni paketi šalju i primaju više puta. Podatkovne pakete primljene više puta treba zanemariti, ali se za svaki od tih paketa mora poslati potvrda.

5.7 Sortiranje i računanje srednje vrijednosti očitanja

Nakon 5 sekundi svaki čvor sortira sva očitanja (vlastita i primljena od drugih čvorova u mreži) u zadnjih 5 sekundi koristeći vektorske i skalarne oznake vremena te ih ispisuje na sučelje. 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.

5.8 Zaustavljanje rada čvorova

Nakon što čvorovi prime **"Stop"** kontrolnu poruku, prvo se zaustavlja UDP komunikacija između čvorova, te se nakon toga sortiraju i ispišu očitani i primljeni podaci. Nakon ispisa se čvor zaustavlja.