FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Raspodijeljeni sustavi - Izvještaj

KREIRANJE STABLA U ASINKRONOJ MREŽI

Toni Sente,
Branimir Škrlec,
Janko Šalković

Zagreb, siječanj 2018.

Sadržaj

Uvod

Kako iz dana u dan broj ljudi koji koriste internet postaje sve veći, pažljivo planiranje i mrežnih sustava i protokola od ključne je važnosti za funkcioniranje ovako velike mreže. Vrlo je bitno smanjiti svaku nepotrebnu redundanciju u sustavu. Slanje poruke u decentraliziranoj mreži vrlo se brzo može pretvoriti u kaos ako radimo tako da svaki korisnik u toj mreži (eng. *peer*) prosljeđuje poruku svim svojim susjedima. Takav način komunikacije generiran izrazito puno nepotrebnog mrežnog prometa. Jedno od poboljšanja koje se može napraviti je da prije slanja poruke izgeneriramo stablo po kojem će poruka biti poslana. Ako krenemo u obilazak stabla iz nekog proizvoljnog čvora tako da se širimo na sve susjede osim na onog od kojeg smo primili poruku, tada ćemo obići sve čvorove točno jednom. Upravo bi na taj način poruka, koju neki čvor želi poslati kroz mrežu od n čvorova, stigla do svih ostalih čvorova točno jednom. Komunikacijska složenost je u ovom slučaju O(n) što je i optimalno rješenje jer moramo obići n čvorova.

U ovom projektu bit će pojašnjeno kako u *peer-to-peer* mreži, s proizvoljnim brojem čvorova, poslati poruku tako da svi čvorovi u mreži prime odgovarajući poruku samo jedanput. Za slanje poruka koristi se TCP protokol. Osim samih čvorova u mreži treba postojati jedan centralni čvor/repozitorij koji ne sudjeluje u razmjeni poruka između ostalih čvorova, već samo održava listu trenutno postojećih čvorova u mreži.

Arhitektura sustava

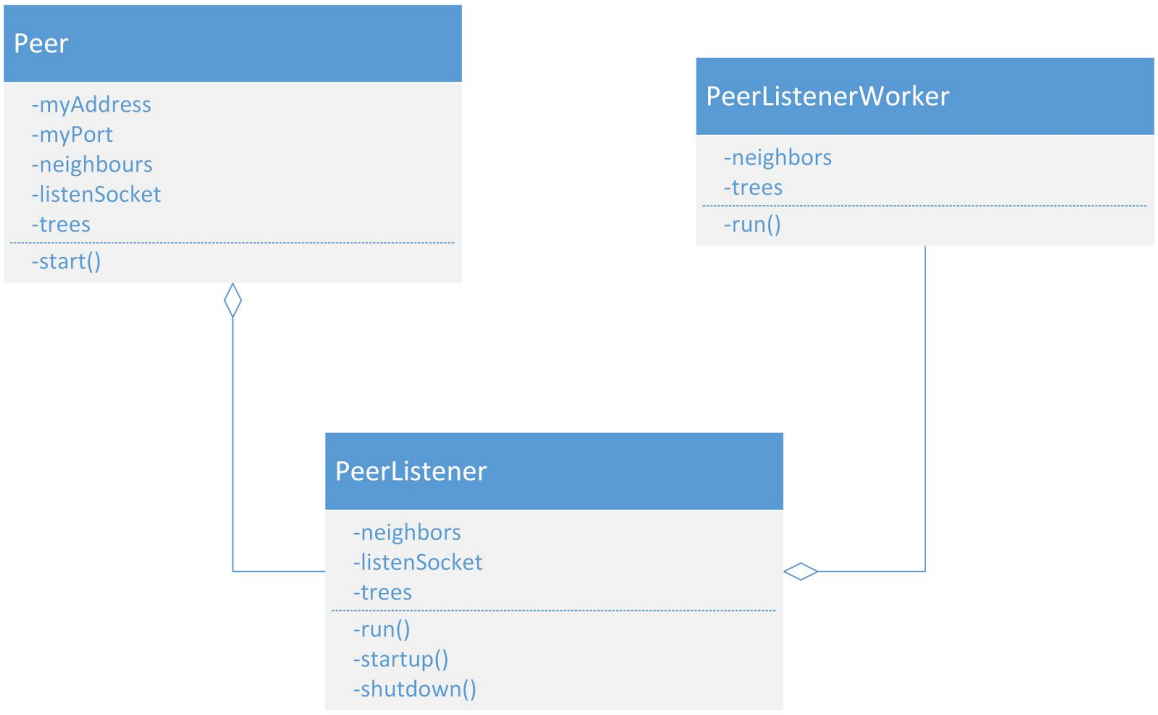
Projekt je napravljen korištenjem programskog jezika Java te se izveden u dva Java projekta – prvi za centralni repozitorij, a drugi za sam mrežni čvor. Oba projekta su napravljena kao konzolne aplikacije. Implementacija čvora je izvedena tako da neke stvari koje su inače automatizirane ovdje je potrebno napraviti pomoću posebnih naredbi kako bi bilo lakše pratiti rad čvora tijekom demonstracije projekta. Rad centralnog repozitorija je relativno jednostavan u usporedbi s čvorom, stoga je njegov rad u potpunosti automatiziran i ne zahtijeva interakciju s korisnikom.

Čvor

Svaki čvor je jedinstveno određen javnom pripadajućom mrežnom adresom koja se sastoji od ip adrese i porta. Čvor je realiziran kao više dretven program koji nakon pokretanja u inicijalizacijskom procesu iz glavne dretve pokreće drugu dretvu čija je svrha slušanje dolazećih poruka s javne adrese, dok glavna dretva ostaje slušati korisnikove naredbe preko naredbenog retka.

Čvor inicijalno nema niti jedne veze prema drugim čvorovima, stoga odmah kontaktira centralni čvor da mu pošalje listu od nekoliko postojećih i aktivnih čvorova u mreži kako bi se s njima postao mrežni susjed. Na isti način, čvor može u bilo kojem trenutku poslati isti zahtjev centralnom čvoru ukoliko se dogodi da njegovi mrežni susjedi ispadnu iz mreže, kako se ne bi dogodilo da čvor postane nepovezan s ostatkom mreže.

Kako bi smanjili količinu poruka koje se šalju kroz mrežu, prije slanja poruke potrebno je imati definirano stablo koji povezuje sve ostale čvorove u grafu, te čiji put između bilo koja dva čvora u tom stablu je jedinstven. Tako svaki čvor prije nego što krene u slanje ili prosljeđivanje poruke, treba imati definiranu listu susjeda u tom stablu kojima treba poslati poruku. Kako bi izbjegli komplikacije s ispadajućim čvorovima u mreži, u našoj implementaciji se prije slanja bilo koje poruke kroz mrežu stvara novo stablo, te se nakon toga kreće sa slanjem same poruke. Detaljnija razrada o stvaranju stabla i slanju poruke opisana je u poglavlju „Implementacija“.



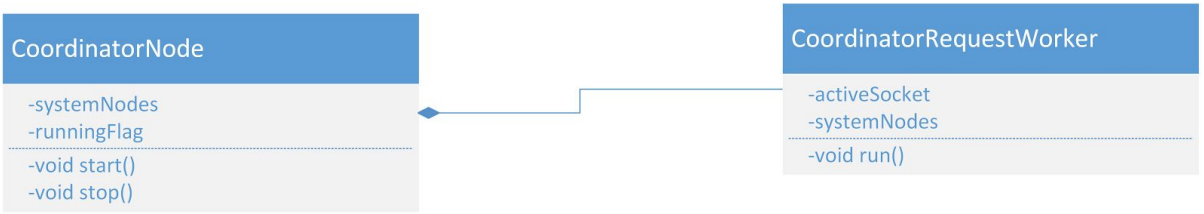
Centralni čvor

Centralni čvor je poseban čvor u sustavu koji ne sudjeluje u razmjeni u poruka među čvorovima, već mu je uloga da samo održava listu postojećih i aktivnih čvorova u sustavu. Mora biti dostupan cijelo vrijeme jer bi njegovim ispadom mreža prestala funkcionirati. Ima javnu mrežnu adresu preko koje svi ostali čvorovi dolaze u kontakt s njime. Isto kao i običan čvor, centralni čvor je realiziran kao više dretven program čija glavna dretva sluša nadolazeće zahtjeve, te sve ih sve prosljeđuje na svoje pomoćne dretve na obradu.

Centralni čvor u strukturi podataka mape drži spremljene sve čvorove koji su trenutno prisutni u sustavu. Prilikom bilo kakve komunikacije s centralnim čvorom, otvara se nova TCP konekcija, a s njom i aktivni soket za komunikaciju sa spojenim korisničkim čvorom. Obrada zahtijeva obrađuje se u novoj dretvi pokrećući metodu run() objekta klase *CoordinatorRequestWorker.* Centralni čvor obrađuje zahtjeve za:

- Registracijom novog čvora
- Dohvaćanjem fiksnog broja čvorova
- Dojavom o ispadu čvora

U nastavku je prikazan dijagram razreda svih klasa povezanih uz sam centralni čvor. Razredi *CoordinatorNode* i *CoordinatorRequestWorker* povezani su vezom agregacije, tj. *Worker* je dio cjeline *CoordinatorNodea*.



Komunikacija u mreži

Komunikacija između svih čvorova u mreži odvija se pomoću TCP protokola. Zbog jednostavnosti, naš projekt radi samo na lokalnom računalu tako da svi čvorovi koriste ip adresu „localhost“, dok se port svakog od čvorova zadaje kao ulazni parametar (program će raditi i ako postavimo da se port određuje sam prilikom pokretanja, no u svrhu lakše demonstracije, mi postavljamo port preko parametra). Čvorovi komuniciraju na preko predefinirani poruka koje su podijeljene u dvije skupine – poruke koje se koriste između centralnog čvora i običnog čvora, te poruke koje definiraju komunikaciju između običnih čvorova u mreži. Poruke su definirane na način da su vrijednosti odvojene znakom „;“. Prva vrijednost u svim porukama označava vrstu poruke, dok ostale vrijednosti/argumenti označavaju vrijednosti ovisne o tipu same poruke.

Komunikacija između centralnog čvora i običnih čvorova

Komunikacija između centralnog čvora i ostalih čvorova u mreži odvija se preko 3 predefinirane poruke:

- Vrsta 1:
  - Poruku vrste 1 se šalje kada se inicijalizira čvor, kako bi ga centralni čvor zapamtio i registrirao na listu postojećih čvorova u mreži
  - Prvi argument predstavlja IP adresu samog čvora
  - Drugi argument predstavlja port samog čvora
  - Izgled stringa (bez navodnih znakova): „1;ip_adresa;port”
  - Napomena: razlog zašto moramo slati IP adresu i port čvora, je taj što za komunikaciju s centralnim čvorom, čvor koristi drugi socket koji ima, vrlo vjerojatno istu ip adresu (localhost), ali drugačiji port(socket koji koristi „javnu adresu i port“ koristi se za slušanje nadolazećih poruka)
- Vrsta 2:
  - Poruku vrste 2 se šalje kada čvor od centralnog čvora traži mrežne adrese n čvorova s kojima bi se mogao spojiti u mreži
  - Prvi argument je n - broj traženih čvorova
  - Izgled stringa (bez navodnih znakova): „2;5“ (zahtjev za 5 mrežnih adresa čvorova koji su aktivni u mreži)
- Vrsta 3
  - Poruku vrste 3 se šalje kada čvor detektira ispad nekog drugog čvora u mreži te javlja centralnom čvoru da je dotični čvor nedostupan
  - Izgled string: “3;ip_adresa;port” (ip adresa i port nedustupnog čvora)

Komunikacija između čvorova

Komunikacija između svih običnih čvorova u mreži definirana je sa četitri predefinirane poruke:

- Vrsta 4
    - Poruka vrste 4 se koristi kada neki novi čvor dolazi u mrežu te se javlja drugim, odabranim, čvorovima za uspostavu veze susjedstva.
    - Kod ove poruke se očekuje odgovor čvora kojemu se šalje zahtjev za susjedstvom
    - Izgled stringa koji se šalje potencijalnom susjedu: „4;ip_adresa_novog_covra;port_novog_cvora“
    - Odgovor koji se očekuje od susjeda je “0” ili “1” (prihvaćeno ili odbijeno susjedstvo)
    - Napomena: u našem programu zbog jednostavnosti susjedstvo uvijek prohvaća
- Vrsta 5
    - Koristi se za stvaranje novog stabla u mreži
    - Ovu poruku generira jedan početni čvor (root) dok ju svi ostali prosljeđuju svojim mrežnim susjedima
    - Izgleda string: “„5;ip_adresa_roota;port_roota;javna_ip_adresa_posiljatelja;javni_port_posilja telja”
- Vrsta 6
    - Poruka vrste 6 koristi se kada neki čvor (root) kroz mrežu želi poslati neku poruku
    - Preduvjet za slanje poruke je da postoji već stvoreno stablo u mreži po kojoj će poruka putovati
    - Izgled string: „6;ip_adresa_roota;port_roota;string (poruka) “
- Vrsta 7:
    - Poruka vrste 7 koristi se za stvaranje odnosa dijete-roditelj tijekom generiranja stabla u mreži
    - Izgled poruke: „7;ip_roota;port_roota;javna_ip_dijeteta;javni_port_dijeteta“

Implementacija

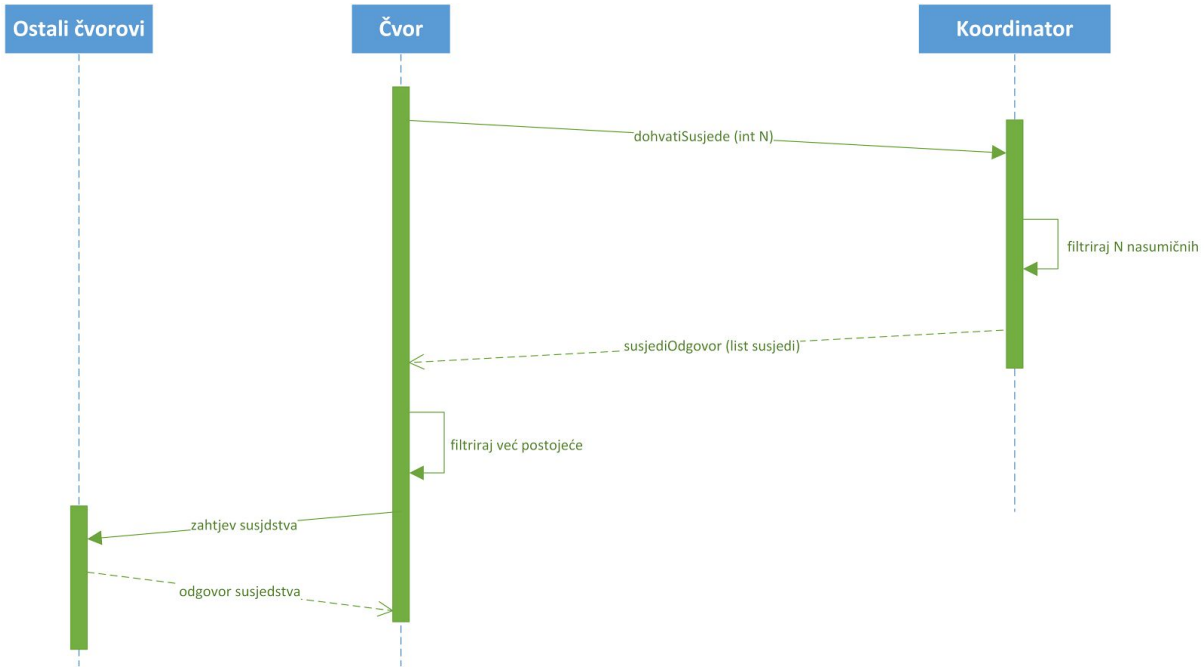
Sekvencijski dijagrami

Sekvencijski dijagrami spadaju u osnovne dijagrame UML standarda. Ostvareni su elementima aktora i njihovih životnih linija, te porukama koje ti aktori izmjenjuju međusobno ili šalju sami sebi. Time se nekoj funkcionalnosti daje komponenta vremenske sljednosti.

Sekvencijskim dijagramima će se u nastavku predočiti osnovni implementacijski mehanizmi i funkcionalnosti.

Zahtjev za mrežnim susjedima

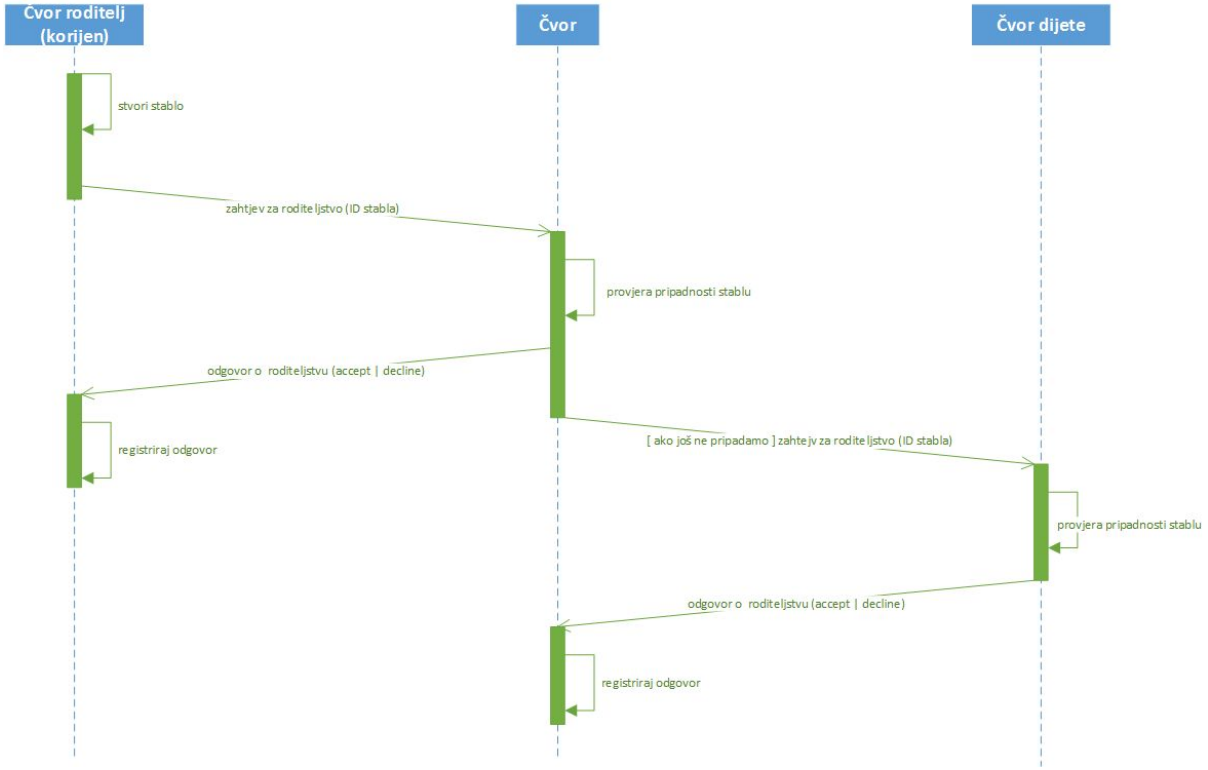
Ako neki čvor u raspodijeljenom sustavu ima nedovoljno susjednih čvorova, on od centralnog čvora može zatražiti određeni broj čvorova. Ta lista čvorova koju dobije može sadržavati i čvorove kojima je on već susjed, te ih stoga filtrira. Iako je odijelio nove susjedne čvorove, mora s njima iskomunicirati susjedstvo odnosno mora im poslati zahtjev za susjedstvo i dobiti odgovor koji može biti potvrdan ili ne-potvrdan. Dobivanjem odgovora, čvorovi susjedi su mu potencijalna djeca u budućim generiranjima stabla. Prikazano na grafu u nastavku.



Stvaranje stabla

Kada neki čvor želi stvoriti stablo za slanje poruke, onda on šalje svoj svojoj djeci zahtjev za roditeljstvo. To uradi u asinkronoj poruci prema čvoru djetetu. Čvor dijete provjerava je li možda već dio tog stabla po nekom drugom čvoru roditelju. Ukoliko je, šalje natrag odgovor s ne-potvrdnom porukom. Ako mu je to prvi roditeljski zahtjev za to stablo, onda taj čvor šalje potvrdan odgovor čvoru koji će mu tim odgovorom postati roditelj. Čvor

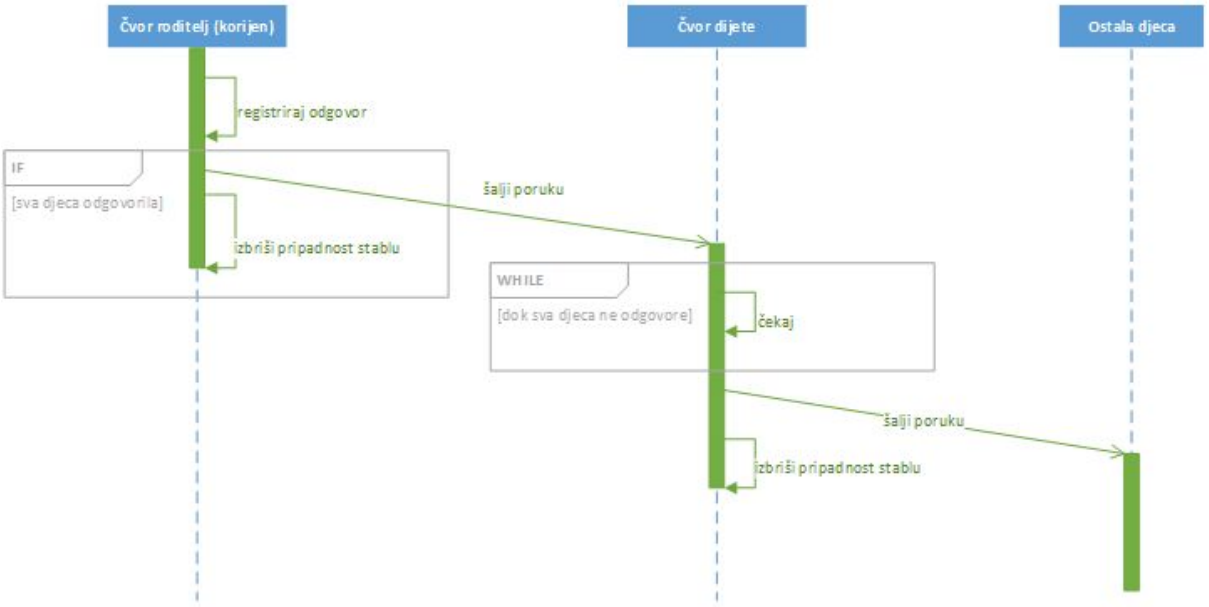
koji je nedavno poslao takve zahtjeve za roditeljstvom, asinkrono čeka odgovore. Ukoliko gledamo sa stajališta korijenskog čvora, on nakon što je primio sve odgovore (bili oni pozitivni ili negativni) može krenuti slati poruke svoj djeci koja su ga potvrdila za roditelja. S druge strane, ako gledamo sa stajališta nekog središnjeg čvora, on nakon potvrđivanja roditelja ne mora imati stvoreno stablo ispod sebe, ali može primiti emitiranu poruku od korijenskog čvora. U takvom slučaju, taj čvor mora čekati dok ne primi sve odgovore od svoje djece, odnosno dok nije stvoreno stablo na sljedećoj razini ispod njega. Proučite dijagram u nastavku.



Propagiranje poruke

Propagacija poruke kreće od korijena stabla. Korijen stabla prilikom primitka potvrde od svojeg djeteta provjerava jesu li mu sva djeca odgovorila. Ako su sva djeca odgovorila, on zna da je stablo stvoreno barem jednu razinu ispod njega, a možda i dalje, te on može poslati poruku svojoj djeci koja su mu odgovorila potvrdno oko roditeljstva.

Neki središnji čvor koji primi poruku svojstvenu nekom stablu, ne smije ju propagirati dalje svojoj djeci sve dok sva ta djeca nisu odgovorila na zahtjev za roditeljstvo (potvrdno ili ne-potvrdno). Nakon što bilo koji čvor pošalje poruku svojoj djeci, on je odradio svoju obavezu prema tom stablu te može maknuti podatke o tome stablu, tj. izbrisati svoju pripadnost tom stablu. Vidljivo na grafu u nastavku.



Detalji implementacije

U nastavku će biti pojašnjeni detalji pojedinih implementacija popraćenih njima svojstvenim programskim kodom.

Čvor

Klasa Peer je klasa koja opisuje karakteristike koje ima svaki čvor u mreži. Svakom čvoru se prilikom pokretanja pridjeljuje jedna dretva koja sluša zahtjeve od susjeda te kreira odgovore, te jedna dretva koju korisnik koristi kako bi unosio naredbe s naredbenog retka o željenoj radnji (ovo se ostvaruje u metodi *start()* prikazanoj na slici ispod).

```
public void start() {
    System.out.println("Starting ... ");

    // start listening tread
    Thread listenThread = new Thread(new PeerListener(neighbours, listenSocket, runningFlag, trees, syncKey));
    listenThread.start();

    // Utils.sleep(500);

    // sayHelloToCoordinator();

    // wait for user input
    waitForCommand();
}
```

Svaki se čvor prilikom uključivanja također mora javiti centralnom repozitoriju kako bi se registrirao. Čvor pritom mora poslati poruku tipa 1 repozitoriju čime ga on sprema u svoju bazu. Čvor za to koristi metodu *sendMessage()* koja će biti opisana nešto dalje u tekstu. Nakon registracije čvor od repozitorija može zahtijevati popis čvorova koji su mu potencijalni susjedi. Ovaj zahtjev opisuje poruka tipa 2. Nakon što od repozitorija dobije popis čvorova, on šalje zahtjev za susjedstvo svim čvorovima s popisa. Zahtjev za susjedstvo opisuje

poruka tipa 4. Metoda *askForNeighborAcceptance(List<String>)* koja ostvaruje ovu funkcionalnost prikazana je na idućoj slici.

```
/*
 * Method used for checking who wants and can be this node neighbor.
 */
private void askForNeighborAcceptance(List<String> nb) {
    String ip = null;
    Integer port = null;
    String message = null;
    String answer = null;
    for (String s : nb) {
        if (!this.neighbours.contains(s) && !s.equalsIgnoreCase(this.MY_ADDRESS + ";" + this.MY_PORT)) {
            ip = s.split(";")[0];
            port = Integer.parseInt(s.split(";")[1]);
            message = "4;" + this.MY_ADDRESS + ";" + this.MY_PORT;
            answer = Utils.messageWithAns(ip, port, message);
            if (!answer.equalsIgnoreCase("1")) {
                continue;
            } else {
                this.neighbours.add(s);
                System.out.println(ip + ";" + port + " zeli biti tvoj susjed!");
            }
        }
    }
}
```

Svaki čvor trenutno može samo prihvatiti odgovor za susjedstvo, no u kasnijim verzijama programa postojat će i mogućnost odbijanja zahtijeva ovisno o broju trenutnih susjeda.

Svaki čvor ima mogućnost poslati poruku kroz mrežu. Poruku koju jedan čvor pošalje kroz mrežu svaki čvor mora primiti točno jednom. U ovom programu to je ostvareno na način da se prije samog slanja poruke kroz mrežu kreira stablo u toj mreži. Metoda *treeRequest()* započinje generiranje stabla slanjem poruke tipa 5 svim svojim susjedima (prikazana na slici ispod).Poruka tipa 5 je svojevrstan zahtjev za “roditeljstvo” kojeg čvor šalje susjedima i na koje prima odgovor. Svako je stablo definirano svojim identifikatorom - u ovom slučaju to je IP adresa i port čvora koji je započeo generiranje stabla.

```
/**
 * Broadcast message for tree creation.
 */
private void treeRequest() {

    String key = MY_ADDRESS + ";" + MY_PORT;
    trees.put(key, new ArrayList<ChildNode>());

    String message = String.format("%d;%s;%s;%s;%s", 5, MY_ADDRESS, MY_PORT, MY_ADDRESS, MY_PORT);

    for (String neighbour : neighbours) {
        trees.get(key).add(new ChildNode(neighbour));
        Utils.sendMessage(neighbour, message);
    }
}
```

Poruka tipa 5 primljena na drugom čvoru inicira spremanje identifikatora stabla u popis trenutno aktivnih stabala, generira odgovor na zahtjev za roditeljstvo čvora od kojeg je primio poruku te nastavlja generiranje stabla u mreži slanjem poruke tipa 5 svojim susjedima. Odgovor na zahtjev za roditeljstvo je poruka tipa 7, a metoda koja obrađuje te odgovore dana je na sljedećoj slici.

```java
// message type 7
private synchronized void acceptParentship(String rootIpAddress, String rootPort, String ipSender, String portSender,
        String status) {
    // String key = rootIpAddress + ";" + rootPort; // TODO ?!
    String key = rootIpAddress + ";" + rootPort;
    // String value = ipSender + ";" + portSender; // TODO ?!
    String sender = ipSender + ";" + portSender;

    for (ChildNode child : trees.get(key)) {
        if (child.getTransportAddress().equalsIgnoreCase(sender)) {
            if (status.equalsIgnoreCase("0"))
                child.setChildStatus(ChildStatus.DECLINED);
            else
                child.setChildStatus(ChildStatus.CONFIRMED);
        }

    }

}
```

Nakon što se završi generiranje stabla, čvor šalje poruku tipa 6 u mrežu. Metoda koji služi za slanje svih gore navedenih poruka je *sendMessage()* te je prikazana u nastavku.

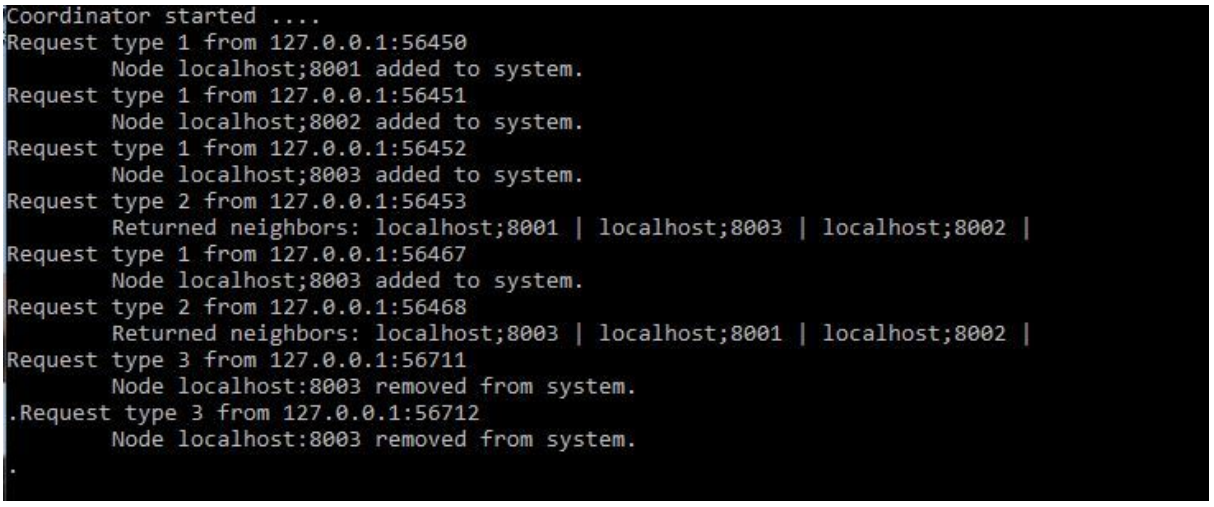
```java
  public static void sendMessage(String destinationIpAddress, Integer destinationPort, String message) {

      try (Socket clientSocket = new Socket(destinationIpAddress, destinationPort)) {

          PrintWriter outToServer = new PrintWriter(new OutputStreamWriter(clientSocket.getOutputStream()), true);

          outToServer.println(message);//WRITE

      } catch (NumberFormatException e) {
          e.printStackTrace();
      } catch (UnknownHostException e) {
          //e.printStackTrace();
          System.err.println("Hitile je exc!");
      } catch (IOException e) {
          e.printStackTrace();
      }
  }
```

Upute za korištenje

Ova aplikacija je konzolna aplikacija, te ju možemo pokrenuti iz bilo koje naredbenog sučelja kojemu smo ugradili java virtualni stroj i java interpreter u putanju (eng. path). centralnog čvora, odnosno čvor pokrećemo sljedećim naredbama:

- "**javac CoordinatorNode.java**" za kompajlanje, te
  "**java CoordinatorNode**" za pokretanje
- "**javac Peer.java**" za kompajliranje, te
  "**java Peer**" za pokretanje.

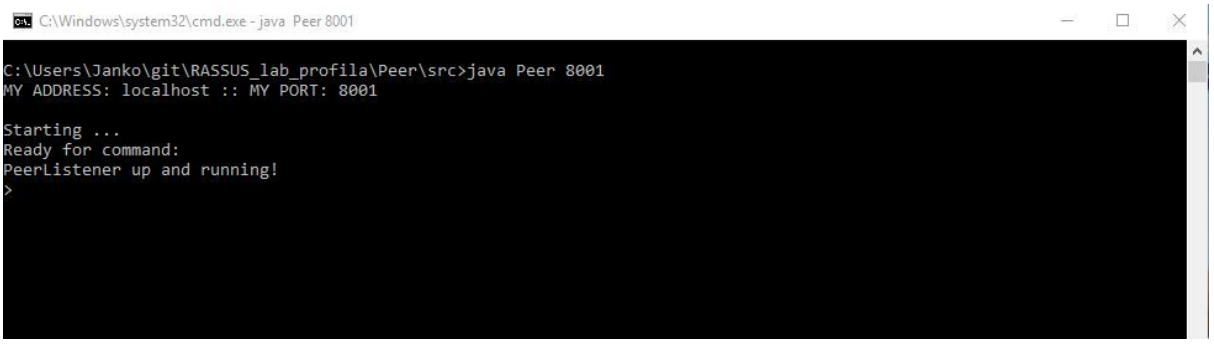
Centralni čvor radi automatizirano te ga je dovoljno samo pokrenuti. Ovisno o primljenim porukama, u konzolnu liniju ispisivat će kontrolne poruke. Poruke će biti finalna potvrda izvršenih radnji na centralnom čvoru. U nastavku je prikazano nekoliko takvih kontrolnih ispisa u jednoj sesiji testiranja aplikacije.



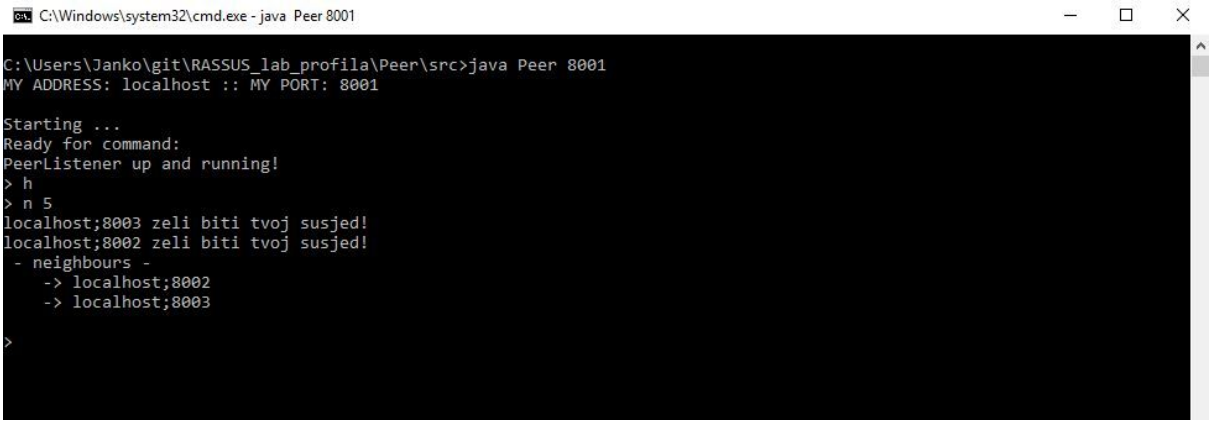
Čvor isto tako radi potpuno automatizirano, ali iz razloga prezentacije rada aplikacije, akcije izvršava samo na unos naredbi u konzolu pokrenutog čvora. Čvor reagira na sljedeće naredbe:

Naredba	Značenje(eng.)	Opis
h	"hello"	Javlja centralnom čvoru da je ovaj čvor ušao u raspodijeljeni sustav.
n <broj_cvorova>	"neighbors"	Naredba koja prima i broj. Dohvaća od centralnog čvora neki fiksni broj čvorova.
a <port>	"add"	Samostalno dodavanje nekog čvora (na temelju porta, na lokalnom računalu) za susjeda.

m	“message”	Slanje poruke.
q	“quit”	Zaustavljanje rada čvora.



Slika iznad pokazuje konzolu aplikacije čvora nakon pokretanja. Aplikacija je u stanju čekanja. U nastavku je unesena komanda za dohvaćanje pet čvorova od centralnog čvora. Nakon što je čvor primio popis susjeda, pokušava s njima ostvariti susjedstvo, što je predočeno kontrolnim ispisom: **“localhost;8003 zeli biti tvoj susjed!”**. Isto tako se ispisuju i svi trenutni susjedi ovog čvora. Dohvaćanje susjeda je nužan korak prije kreiranja stabla i slanja poruke. Detalji razmjena poruka kod dohvaćanja susjeda opisani su sekvencijskim dijagramom u odjeljku nešto više u dokumentu.



Sve je spremno za slanje poruke, te je naredbom **m** možemo i inicirati. Da se prisjetimo, slanje poruke zahtijeva prvotno kreiranje stabla. Završetak kreiranja stabla predstavljeno je kontrolnim ispisom prikazanim na slici u nastavku.

Što se tiče nekog središnjeg čvora, on nakon što primi poruku i ispiše da ju je primio, pokušava propagirati tu poruku svojim čvorovima djeci, ukoliko ih ima.

Kontrolni ispis primljene poruke u sebi uvijek sadrži adresu korijenskog čvora koji je inicirao slanje poruke. Ukoliko čvor nema djece, neće ni pokušati stvoriti stablo ispod sebe te kontrolna poruka o stvaranju stabla niti neće biti ispisana.

Ukoliko za vrijeme izgradnje stabla ili potvrđivanja susjedstva dođe do ispada nekog čvora, čvor koji je zamijetio ispad dužan je centralnom čvoru javiti o primjećenom propustu. Takav propust predstavljen je kontrolnom porukom **localhost;8003 is not present. Informing coordinator… .** Proučite sliku u nastavku.

Na samim korisnicima je da istestiraju ovu aplikaciju do kranjih mogućnosti. Moguća je pojava prestanka rada sustava. Ova aplikacija je još u fazi razvoja te joj predstoji dosta iteracija unaprjeđivanja.