Diplomski studij

Informacijska i komunikacijska tehnologija
Telekomunikacije i informatika

Računarstvo
Računarska znanost
Programsko inženjerstvo i informacijski sustavi

Raspodijeljeni sustavi

Upute za izradu 1. domaće zadaće
Raspodijeljeno programiranje (TCP, UDP i RMI)

Ak. g. 2013./2014.

Sadržaj

1 Uvod

CILJ DOMAĆE ZADAĆE:

U praksi utvrditi i ponoviti gradivo s predavanja. Studenti će naučiti programirati raspodijeljeni sustav temeljen na modelu klijent-poslužitelj koristeći 3 izvedbe komunikacije (protokoli TCP i UDP te poziv udaljene metode – RMI).

ZADATAK:

Ova domaća zadaća se sastoji od sljedeća 3 dijela:

- **0. proučavanje primjera s predavanja**,
- 1. programiranje poslužitelja raspodijeljenog sustava,
- 2. 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:

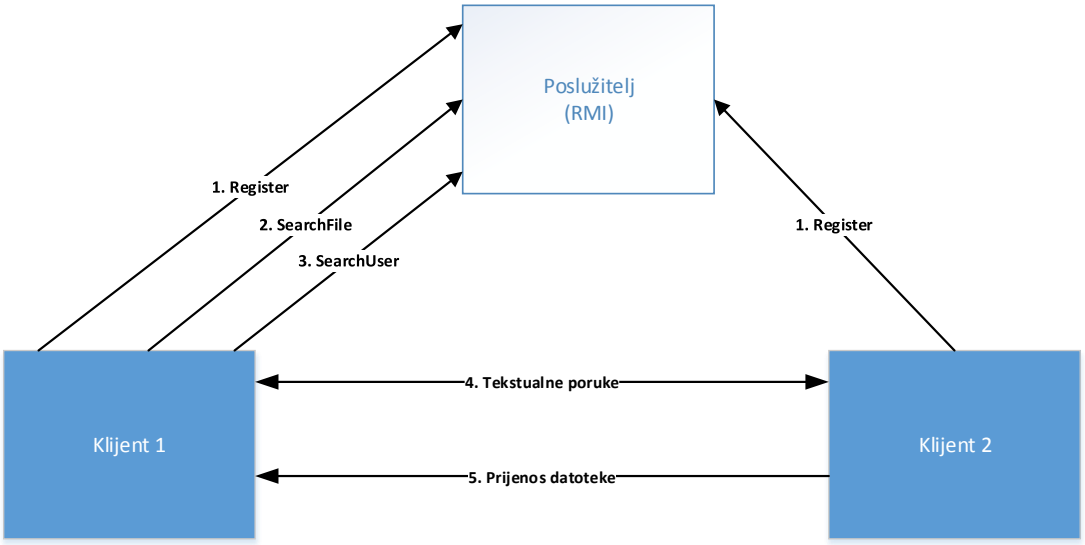
1. projekt klijenta,
2. projekt poslužitelja

Navedene komponente trebaju biti realizirane u nekom od objektno-orijentiranih programskih jezika kao što su Java, C++, C# itd. Projekt je skup datoteka u odabranom razvojnom okružju (npr. Eclipse, Netbeans, Visual Studio, itd.).

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.

2 Arhitektura raspodijeljenog sustava



Raspodijeljeni sustav se sastoji od jednog poslužitelja i više instanci klijenata. Klijent se prilikom pokretanja registrira kod RMI poslužitelja, a nakon toga pozivom odgovarajuće metode može pretražiti dijeljene datoteke ostalih klijenata ili zatražiti podatke o adresi i vratima na kojima može kontaktirati drugog klijenta. Komunikaciju tekstualnim porukama je potrebno ostvariti koristeći protokol TCP, a transfer datoteka koristeći protokol UDP. Klijent treba koristiti ista vrata za slušanje dolaznih poruka za protokole UDP i TCP.

Napomena: Prilikom ostvarivanja ove tri vrste komunikacije posebnu pozornost obratite na to da klijent i poslužitelj ne smiju biti realizirani u istom projektu odabrane razvojne okoline, ali zato projekti trebaju dijeliti zajednička sučelja.

3 UDP komunikacija između dva klijenta

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 klijenti (Klijent 1 i Klijent 2) 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`. Ova klasa se koristi 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 gubitaka paketa u mreži [postotak] i prosječno kašnjenje paketa u jednom smjeru [milisekunde].

Napomena:Ukoliko se student odluči za neki drugi programski jezik osim Jave, tada će morati simulirati stvarnu mrežu po uzoru na način ostvaren klasom `SimpleSimulatedDatagramSocket`.

Klijent 1 treba zahtijevati dostavu odgovarajuće podatkovne datoteke od Klijenta 2. Klijent 2 će datoteku podijeliti u nekoliko paketa, numerirati pakete i **poslati ih klijentu jedan za drugim (odjednom) te zatim čekati potvrde od svih poslanih paketa**. **Prilikom slanja paketa, Klijent 2 treba koristiti jedan jedinstveni spremnik veličine 10KB (ograničava maksimalnu veličinu paketa).** Klijent 1 treba voditi brigu o redoslijedu primljenih paketa i sve dolazne pakete stavljati u spremnik veličine datoteke. Kada Klijent 1 primi sve pakete, datoteku će pohraniti na tvrdi disk vašeg računala. Za svaki uspješno primljeni paket, Klijent 1 će Klijentu 2 poslati potvrdu (također u obliku paketa). Klijent 2 treba voditi brigu o izgubljenim paketima te ponoviti slanje svih izgubljenih paketa. Moguća je i situacija u kojoj će poneki paket biti poslan i po nekoliko puta.

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

4 TCP komunikacija između dva klijenta

Prilikom izrade domaće zadaće potrebno je maksimalno iskoristiti programski kod s predavanja te programski izvesti tekstualnu komunikaciju protokolom TCP. Klijenti moraju biti u mogućnosti održavati više konkurentnih sjednica, tj. istodobno komunicirati s više klijenata. Kada jedan od sudionika napusti komunikaciju, drugom se korisniku treba ispisati odgovarajuća poruka.

5 RMI komunikacija između klijenta i poslužitelja

Prilikom izrade domaće zadaće maksimalno iskoristite programski kod s predavanja te programski izvedite komunikaciju klijenta i poslužitelja koji komuniciraju pozivom udaljene metode. Poslužitelj treba implementirati 3 metode. Jedna metoda će služiti za registraciju klijenta kod poslužitelja (**`boolean register(String username, String sharedFiles, String address, int port)`**), druga metoda pretražuje dijeljene datoteke od klijenata spojenih na poslužitelj i kao rezultat vraća popis klijenata koji sadrže traženu datoteku (**`String searchFile(String filename)`**), a treća metoda vraća klijentsku adresu i vrata na kojima klijent sluša **sve** dolazne zahtjeve (**`UserAddress searchUser(String username)`**).

Klase koja implementiraju navedena sučelja potrebno je ostvariti na poslužiteljskoj strani.

Klijent treba sve klase potrebne za komunikaciju s poslužiteljem dohvaćati putem Web poslužitelja Tomcat (ili nekog drugog, npr. Internet Information Services) kojeg trebate prethodno instalirati, podesiti i pokrenuti. Web poslužitelj će stoga obavljati ulogu poslužitelja klasa (koju je u primjeru s predavanja obavljala klasa `ClassServer`).

Redoslijed pokretanja RMI primjera je:

1. Pokretanje RMI registra:

   ```
   C:\Program Files\Java\jdk1.6.0_16\bin\rmiregistry.exe,
   ```

2. Pokretanje poslužitelja klasa (Tomcat, IIS),
3. Unošenje parametra Java virtualnih strojeva (u NetBeans – desni klik na projekt -> Properties -> Run -> VM Options):

   ```
   -Djava.rmi.server.codebase=http://localhost:4727/
   -Djava.security.manager -Djava.security.policy=myclient.policy

   -Djava.rmi.server.codebase=http://localhost:4727/
   -Djava.security.manager -Djava.security.policy=myserver.policy
   ```

4. Pokretanje RMI poslužitelja i
5. Pokretanje RMI klijenta.