RafBook dokumentacija

Mihajlo Madžarević 55/20RN

1. Uvod

RafBook je projekat namenjen primeni paralelne komunikacije čvorova u svrhu rada sa ASCII kodiranim podacima. Projekat se fokusira na primenu paralelne sinhronizacije komunikacije procesa, kao i same sinhronizacije rada unutar procesa. Projekat se bazira na CHORD čvornom sistemu. Sinhronizacija između čvorova se realizuje upotrebom token sistema algoritmom Suzuki-Kasami.

2. Pokretanje sistema

Čvorovi komuniciraju na mreži i za njihovo podizanje neophodno je pružiti tekstualni dokument pod nazivom servent_list.properties i smestiti ga pod direktorijum chord koji se nalazi u glavnom radnom direktorijumu projekta. U daljem tekstu pojam čvora može se pronaći i pod nazivom servent.U samom dokumentu treba pružiti sledeće parametre:

1. root – absolutna putanja do direktorijuma sa ASCII dokumentima
2. ip – ip mreža na kom će sistem funkcionisati (za naše potrebe localhost)
3. weak_limit – slaba granica vremenske detekcije otkaza čvora u milisekundama
4. strong_limit – jaka granica vremenske detekcije otkaza čvora u milisekundama
5. servent_count – broj servenata u sistemu
6. chord_size – veličina CHORD sistema za skladištenje podataka i čvorova
7. bs.port – port bootstrap servera o kojem će biti reči u daljem tekstu
8. servent#.port – port na kome će server pod id-jem # osluškivati

Da bismo pružili novi servent u sistem moramo definisati njegov port u ovom dokumentu.

Svi parametri se pružaju sa znakom jednako (=) bez razmaka. Primera radi da bismo dodali novi čvor na port 1400 pod id-jem 3 u dokument bismo dodali pravilo (servent3.port=1400).

3. Bootstrap server

Bootstrap server se pokreće prvi prilikom paljenja sistema i odgovoran je za dodavanje svih ostalih servenata u sistem. Bootstrap server nije svestan arhitekture sistema i ne sme komunicirati sa serventima posle uspešnog njihovog dodavanja u sistem. Port serventa koji treba dodati se čita iz dokumenta pomenutog u prethodnom poglavlju, a servent na koji se dodaje ovaj novi servent bira Bootstrap server nasumično iz liste već pokrenutih servenata.

4. Servent

Serventi iliti čvorovi su ključne komponetne sistema. Komuniciraju putem internet poruka i svaki od njih ima osluškivač namenjen za tu komunikaciju. Prilikom pokretanja serventa neophodno je pružiti sistem za iščitavanje pomenutog dokumenta iz poglavlja 2. i skladištiti

date podatke u radnu memoriju prilikom izvršavanja. Nakon iščitavanja dokumenta servent_listener.properties datoteke, servent bi trebalo da pokrene osluškivač poruka, parser komandi i servent pokretač.

5. Osluškivač poruka

Namena osluškivača poruka je da bude pokrenut sve dok servent ne dobije naredbu o zaustavljanju. Tokom rada osluškivač osluškuje poruke formata koji će biti objašnjen u daljem tekstu i pruža opsluživač poruke za svaki definisani tip poruke. Ukoliko tip poruke nije prepoznat to sistem treba da nagovesti korisniku.

6. Parser komandi

Namena parsera komandi je da isčitava komande date u pruženom tekstualnom dokumentu. Svaka komanda se zadaje u novom redu i neophodno je pružati samo one komande koje su već definisane u sistemu. Po potrebi mogu se dodavati nove komande a za njih je neophodno da naslede interfejs koji pruža sledeće metode:

1. String commandName() – vraća ime komande koje će se koristiti u dokumentu za zadavanje komandi serventu
2. void execute(String args) – obrađuje zahtev komanda, argument args su argumenti koji se mogu pružiti komandi

Primer upotrebe komande je get_file 25.txt, gde je get_file ime komande koju funkcija commandName() vraća, a 25.txt je argument args koji se prosleđuje komandi execute(String args) koja dalje taj zahtev obrađuje. Svi odgovori na komande su dati u poglavlju sa Porukama.

7. Servent pokretač

Servent pokretač je zaslužan za kačenje na već postojeći servent u sistemu ukoliko takav servent postoji. Servent pokretač treba da kontaktira Bootstrap server kako bi dobio port na koji treba da se nakači. Servent ovo čini tako čto šalje običnu poruku sa tekstualnim sadržajem:

“Hail <port>\n”

port – port na kojem servent koji treba dodati u sistem sluša.

Bootstrap server će da vrati odgovor:

-2 ukoliko je došlo do greške u komunikaciji

-1 ukoliko je servent koji se dodaje prvi u sistemu

Neki port ukoliko postoje serventi i dobili smo port serventa koji treba da doda novi čvor u sistem. Serventu koji dodaje novi sevent treba poslati poruku NewNodeMessage() o kojoj će biti reči kasnije. Posle komunikacije sa Bootstrap serverom čvor ovde startuje funkcionalnost koja bi trebala da na svakih pet sekundi proveri da li je njen prethodnih (koji će mu biti dodeljen prilikom dodavanja u sistem) funckionalan i sluša na portu kome treba da sluša.

8. Poruke

Sve poruke nasleđuju BasicMessage poruku koja podrazumeva da u konstruktor se prosleđuju tip poruke, port od kojeg se šalje poruka i port kome se šalje poruka, i radi jednostavnosti se neće pisati u konstruktor svakog dalje navedenog tipa poruke. Tipovi poruka koje serventi razmenjuju i njihove namene su sledeće:

1. AddFriendMessage(boolean) – poruka za dodavanje stranog čvora kao prijatelja, argument boolean se postavlja na false prilikom prvog sljanja, prilikom prijema sa strane stranog čvora, argument se postavlja na true i zatim šalje nazad prvobitnom čvoru. Čvor prilikom primanja poruke dodaje čvor koji je poslao poruku u listu prijatelja i ukoliko je boolean argument postavljen na false šalje AddFriendMessage(boolean) sa argumentom postavljenim na true nazad čvoru koji je prvobitno poslao poruku.
2. AllUpdatesDoneMessage() – služi kao sinhrona potvrda da su svi čvorovi obavili ažuriranje svoje tabele sledbenika prilikom dodavanja novog čvora i da čvor koji je dodavao novi čvor može da preda token dalje nekom čvoru.
3. AskGetMessage(String) – služi prilikom dohvatanja ASCII tekstualne datoteke, da se pita naredni čvor po CHORD sistemu da li sadrži datoteku koju tražimo. String argument koji prosleđujemo je kluč (chordID po CHORD sistemu) datoteke koju tražimo. Ukoliko čvor koji primi poruku ima dati ključ, ispisuje ga na konzolu zajedno sa fajlom.
4. BackupMessage(BackupData) – poruka šalje kopiju sistema datoteka čvora koji šalje ka čvoru koji prima poruku. BackupData argument je klasa koja sadrži sledeće informacije o čvoru:
   - int port – port čvora čija se kopija pravi
   - int sequenceNumber – sekvencijalni broj zahteva za čuvanje kopije podataka, čuva se kako bismo znali da li ovu kopiju treba čuvati ili smo već dobili noviju kopiju, tačnije kopiju sa većim sekevncijalnim brojem
   - ServentInfo predecessorInfo – podaci o čvor prethodniku čvora koji je poslao poruku
   - Map<Integer, MyFile> – sistem datoteka koji se kopira gde je Integer ključ (ime čvora, primer 25.txt, kluč bi bio 25) čvora čiji je fajl i MyFile objekat koji sadrži sam fajl i tip čuvanja (privatni ili javni, privatne fajlove mogu da vide samo naši prijatelji dodati sa AddFriendMessage(boolean), javne mogu svi).

   Čvor koji primi ovu poruku treba da sačuva ove podatke u svoj sistem fajlova.

5. FailureUpdatesMessage(ServentInfo) – Slično AllUpdatesDoneMessage() ali prilikom nasilnog zaustavljanja čvora. Ukoliko čvor je nasilno zaustavljen njegov naslednik vidi da čvor ne odgovara na Ping() poruku, a pritom kontaktira prethodnika (šalje mu IsReallyDeadMessage(ServentInfo) poruku) čvora koji je prinudno zaustavljen, ukoliko odgovor ne usledi sistem se rekonstruiše. Čvor naslednik čvora koji je prinudno zaustavljen briše čvor koji je prinudno zaustavljen iz tabele sledbenika i potom šalje svim ostalim čvorovima FailureUpdatesMessage(ServentInfo) poruku dok drži token. Kada svi završe svoje ažuriranje, sistem može da nastavi sa normalnim radom.

6. IsReallyDeadMessage(ServentInfo, int, boolean, boolean) - poruka koju čvor naslednik čvora koji je prinudno ugašen šalje čvoru prethoniku čvora koji je prinudno ugašen da potvrdi da je čvor zaista ugašen. Ukoliko odgovor ne usledi u roku od vremena nagoveštenog u servent_list.properties datoteci pod parametrom strong_limit, čvor se uklanja i sistem rekonstruiše. Redosled slanja je sledeći čvor naslednik šalje ovu poruku čvoru prethodniku sa oba boolean parametra postavljena na false, a int stavlja da bude port čvora koji je sumnjiv da se prinudno zaustavio. Prethodnik prima poruku i zatim usmerava je ka sumnjivom čvoru, postavlja prvi boolean na true, drugi na false, a int postavlja za port čvora naslednika. Sumnjivi čvor ukoliko primi poruku vraća je nazad čvoru prehodniku sa prvim boolean postavljenim na false a drugim na true i portom od čvora naslednika. Čvor prethodnik ovog puta kada primi poruku prosleđuje je nazad čvoru nasledniku sa oba booleana postavljena na true i int portom čvora koji je sumnjiv. Čvor naslednik kada primi ovakvu poruku skida marku sumnjivog čvora. Ukoliko se odziv od sumnjivog čvora ne desi u roku od postavljanja marke plus vremena jake granice otkaza date u servent_list.properties datoteci, čvor se uklanja i počinje rekonstrukcija sistema.
7. ListFilesMessage(List<MyFiles>) – poruka koja se šalje čvoru da izlistamo njegove datoteke. Ukoliko smo prijatelj čvoru kome šaljemo poruku možemo videti sve njegove datoteke, ukoliko nismo možemo videti samo javne datoteke. Poruka se šalje sa null argumentom List<MyFiles> ukoliko tražimo da nam čvor izlista njegove datoteke, a pritom nam on uzvraća ListFilesMessage(List<MyFiles>) gde je argument List<MyFiles> lista fajlova tog čvora.
8. NewNodeMessage() – poruka se šalje od strane servent pokretača kada Bootstrap servent nam vrati port čvora koji treba da doda novi čvor, tom prilikom čvor koji treba dodati šalje ovu poruku čvoru koji treba da ga doda. Prilikom prijema ove poruke čvor koji dodaje novi čvor traži token, kada ga dobije dodaje novi čvor, menja tabelu sledbenika i potom šalje svim ostalim čvorovima u sistemu poruku (UpdateMessage()) da ažuriraju tabelu sledbenika. Kada se ovaj proces završi i čvor koji dodaje novi čvor dobije potvrdu porukom AllUpdatesDoneMessage(), pruža dalje token čvoru kome je to prvo bilo potrebno i sistem nastavlja sa normalnim radom.
9. PingMessage() – poruka koju čvor šalje svom prethodniku da proveri da li se prinudno zaustavio, ukoliko čvor prethodnik vrati odgovor (poruku PongMessage()) znači da se nije prinudno zaustavio.
10. PongMessage() – poruka koju čvor prethodnik šalje svom sledbeniku da potvrdi da se nije prinudno zaustavio i idalje sluša na pretviđenom portu.
11. PutMessage(int, MyFile) – ukoliko neki čvor pokuša da doda ASCII datoteku čiji ključ ne pripada(po CHORD sistemu) tom čvoru, on pronalazi ključ čvora kome ova datoteka najverovatnije pripada i šalje ovu poruku. Prilikom prijema čvor radi istu proveru kao i prethodni čvor i dodaje čvor ukoliko njemu pripada čvor datoteke ili šalje ovu poruku iznova nekom daljem čvoru. Argument int je ključ datoteke a MyFile objekat već prethodno spomenut same datoteke.
12. RemoveFileMessage(int) – na sličan način se dešava provera ukoliko mi sadržimo datoteku pod datim ključem briše je ili u suprotnom traži čvor kojem treba poslati ovu poruku kako bi obrisao datu datoteku.

13. SendMeYourBackupMessage() – poruka koja zahteva od čvora da nam pošalje kopiju njegovih podataka. Koristi se prilikom dodavanja novog čvora kako bi posle zvih poruka ažuriranja, tačnije tačno pre sljanja AllUpdatesDoneMessage() poslali ovu poruku našem prethodniku kako bismo vodili računa o njegovim podacima ukoliko se prinudno zaustavi. Podatke našeg sledbenika smo već prethodno zabeležili jer nas je on dodao. Čvor prilikom prijema ove poruke nam šalje BackupMessage().
14. SorryMessage() – poruka koja se šalje prilikom dodavanja novog čvora. Ukoliko već postoji čvor sa istim chordID-jem (po CHORD sistemu) čvor koji treba da doda novi čvor mu vraća ovu poruku. Po prijemu ove poruke čvor se gasi.
15. TellGetMessage(int, MyFile) – ukoliko je neki čvor tražio ASCII datoteku od nekog drugog čvora i taj čvor sadrži tu datoteku, vratiće mu ovu poruku sa datotekom u MyFile argumentu, a ključ datoteke je dat argumentom int. Po prijemu ove poruke čvor će ispisati ključ i datoteku na konzolu.
16. TokenMessage(Token) – poruka koja prosledjuje token. Token argument je objekat koji sadrži sledeće parametre:
    - Map<Integer, Integer> tokenRequests – vektorski sat koji vodi računa ko je zahtevao token prilikom izvršavanja kritične sekcije
    - Queue<Integer> queue – red čekanja za token, red se sastoji iz portova čvorova koji trebaju redom sledeći da prime token
    Implementacija tokena je po Suzuki-Kasami algoritmu
17. TokenRequestMessage(int, int) – poruka koja se šalje kada servent hoće da uđe u kritičnu sekciju na nivou celog sistema. Prvi argument int je id čvora koji šalje zahtev, neophodan je kako bismo zabeležili koji čvor pravi zahtev u Suzuki-Kasami algoritmu. Drugi argument je koji zahtev po redu pravimo za token sa tog čvora. Po prijemu servent proverava da li ima token i da li nije zastareo zahtev po Suzuki-Kasami algoritmu, a zatim ukoliko su ova dva uslova ispunjena i servent nije u kritičnoj sekciji šalje token serventu koji ga traži. U slučaju da jeste u kritičnoj sekciji, čvor markira zahtev za token čvora u svom vektorskom satu i po izlasku iz kritične sekcije pravi red po kojem bi ostali čvorovi trebalo da prime token (Suzuki-Kasami algoritam).
18. UpdateMessage() – poruka koja se šalje svim čvorovima prilikom dodavanja nekog čvora u sistem. Čvorovi moraju da budu obavešteni kako bi ažurirali svoje tabele sledbenika. Pri prijemu poruke čvor ažurira tabelu sledbenika i šalje sledećem čvoru istu poruku. Kada poruka dođe nazad do čvora koji ju je prvi poslao on šalje poruku AllUpdatesDoneMessage čvoru koji ga je dodao. Poruku inicira čvor koji je dodat prilikom prijema WelcomeMessage poruke.
19. WelcomeMessage(Map<Integer, MyFile> values) – poruka koja se šalje čvoru koji se dodaje od čvora koji ga dodaje. Pri prijemu čvor započinje krug UpdateMessage() poruka koje kruže po sistemu a zatim se vraća na probitni čvor.