RafBook

Projekat

1 Pregled zadatka

Realizovati funkcionalan distribuirani sistem koji će da obezbedi rad sa ASCII kodiranim tekstualnim datotekama i slikama. Sistem korisniku omogućava sledeće:

- Dodavanje nove datoteke sa jedinstvenim nazivom i putanjom u sistem.
- Dohvatanje proizvoljne datoteke iz distribuiranog sistema.
- Dodavanje prijateljskih čvorova.
- Brisanje datoteke, na lokalnom serventu.
- Topološka organizacija sistema omogućava brže dohvatanje datoteka.
- Otpornost na otkaze.

Projekat može da se implementira u proizvoljnom programskom jeziku, dok god zadovoljava sve funkcionalne i nefunkcionalne zahteve. Dozvoljeno je koristiti distribuirani sistem sa vežbi kao polaznu tačku. Da bi bili dodeljeni bilo kakvi poeni, neophodno je da implementirane funkcionalnosti rade na distribuiranom sistemu.

Funkcionalni zahtevi za sistem su opisani u odeljku 2.

Nefunkcionalni zahtevi za sistem su opisani u odeljku 3.

Bodovanje zadatka, kao i instrukcije za predaju zadatka su dati u odeljku 4.

2 Funkcionalni zahtevi

2.1 Osnovne funkcionalnosti

RafBook sistem služi da obezbedi rad sa ASCII kodiranim tektualnim datotekama i slikama. Datoteke treba da budu organizovane u virtuelnoj strukturi datoteka, gde na svakom pojedinačnom čvoru može da bude prisutan proizvoljan podskup ove čitave strukture.
Interakcija sa sistemom se obavlja preko komandne linije (CLI). Virtuelni sistem datoteka treba da se

čuva dok god postoji barem jedan aktivan čvor. Kada se poslednji čvor ugasi, time prestaje da postoji i virtuelni sistem datoteka, i ne treba da bude moguće rekonstruisati ga naknadno.

Osnovne funkcionalnosti za sistem uključuju:

- Dodavanje čvora u listu prijatelja
- Dodavanje nove datoteke u mrežu sa opcijom privatnog ili javnog deljenja.
- Pregled javnih i privatnih datoteka od prijatelja ili nekog drugog čvora.
- Uklanjanje datoteke sa mreže.

Na lokalnom sistemu treba da postoji direktorijum koji je na početku prazan, i koji se koristi pri radu sa sistemom:

- Radni koren - gde će se nalaziti datoteke sa kojima korisnik želi aktivno da radi.

2.2 Konfiguracija čvora

Pri pokretanju čvora, automatski se iščitava konfiguraciona datoteka u kojoj se navode sledeći atributi:

- Radni koren - putanja na lokalnom sistemu gde se skladište datoteke za rad. (string)
- Port na kojem će čvor da sluša. (short)
- IP adresa i port bootstrap čvora - odeljak 3.1. (string i short)
- Slaba granica otkaza - odeljak 3.2. (int)
- Jaka granica otkaza - odeljak 3.2. (int)

Pretpostavlja se da će svi čvorovi imati usklađene konfiguracione datoteke, i nema potrebe dodatno proveravati da li je to slučaj. Dozvoljeno je da sistem ne funkcioniše usled nepravilno podešene konfiguracione datoteke.

2.3 Komande

Korisnik može da zada sledeće komande sistemu:

- add_friend [adresa:port] - Dodavanje čvora u listu prijatelja
- add_file [path] [private/public] - Dodavanje nove datoteke u mrežu sa opcijom privatnog ili javnog deljenja.
- view_files [adresa:port] - Pregled javnih i privatnih datoteka od prijatelja ili nekog drugog čvora.

- remove_file [filename] - Uklanjanje datoteke sa mreže.
- stop - Uredno gašenje čvora.

Kod svih komandi se pri navođenju naziva datoteke očekuje putanja relativna u odnosu na radni koren, koji je naveden u konfiguracionoj datoteci. Nazivi datoteka nikada neće imati razmake, i nema potrebe podržavati ih.

3 Nefunkcionalni zahtevi

3.1 Arhitektura sistema

Dozvoljeno je da postoji bootstrap server, koji nije čvor u mreži (tj. sve napomene u ovom dokumentu koje se odnose na čvorove se ne odnose na bootstrap server). Za bootstrap važe sledeće pretpostavke:

- Koristi se isključivo za prvo uključivanje čvora u mrežu. Čim bootstrap prosledi novom čvoru adresu i port nekog čvora iz sistema, komunikacija sa bootstrap-om se prekida.
- Bootstrap server ima veoma ograničen protok. Komunikacija sa bootstrap serverom mora biti svedena na minimum.
- Nije dozvoljeno da bootstrap server bude svestan arhitekture sistema, te da on bude taj koji će je organizovati. Sistem mora da bude samoorganizujući.

Treba da bude moguće uključivati i isključivati čvorove u bilo kom trenutku rada sistema, uključujući dok se drugi čvorovi uključuju ili isključuju.

Postoje dve varijante za arhitekturu sistema koje se različito boduju:

- **Prost graf (100% poena)** - pošto graf nije kompletan, da bi čvor A prosledio poruku čvoru B, on mora da pronađe (ne nužno kompletnu) putanju kroz sistem do čvora B. Ovde je neophodno da broj skokova između A i B teži logaritamskoj zavisnosti od ukupnog broja čvorova. Ako broj skokova između proizvoljnih A i B teži linearnoj zavisnosti, implementacija se boduje kao da je

rađen kompletan graf (50% poena). Da bi se graf računao kao prost, broj suseda za sve čvorove mora da ima logaritamsku zavisnost od ukupnog broja čvorova. Ne sme da postoji centralna tačka otkaza (čvor nakon čijeg stopiranja sistem prestaje da radi). Ne sme da postoji bottleneck - bottleneck za potrebe ovog projekta definišemo kao čvor (ili više čvorova kojima je fiksan broj) kroz koji komunikacija često teče. Ako postoje čvorovi kroz koje komunikacija često teče, ali njihov broj zavisi od broja čvorova u sistemu, tako da se komunikacija prirodno raspodeljuje među njima, onda oni nisu bottleneck. Startovanje novih čvorova, kao i stopiranje aktivnih čvorova može i treba da prouzrokuje restruktuiranje sistema. Primeri: Chord, Kademlia, Pastry, sopstvena struktura sistema

- **Kompletan graf (50% poena)** - svaki čvor je povezan sa svakim drugim čvorom. Komunikacija je uvek direktna.
-

3.2 Detektovanje otkaza

Otkazivanje čvora se detektuje u dve faze. Kao konstantan parametar sistema treba da postoje slaba granica otkaza (npr 4000) i jaka granica otkaza (npr 10000ms), obe date kao vreme koje se čvor ne javlja na ping.

Ako čvor prekorači slabu granicu otkaza, markiramo ga kao sumnjivog, ali ne započinjemo uklanjanje čvora i restrkutiranje sistema, već tražimo nekom drugom stabilnom čvoru da nam on potvrdi da je sumnjivi čvor zaista problematičan. Ako dobijemo potvrdu da je čvor sumnjiv, i nakon toga istekne jaka granica otkaza, čvor se eliminiše iz sistema i započinje se restruktuiranje.

Kada čvor otkaže, njegov dotadašnji posao ne sme da se izgubi. Ako postoji slobodan čvor (jedan on njegovih buddy-a), on treba da preuzme podatke za koje je bio zaduzen cvor koji je otkazao.

Sistem treba da bude sposoban da se izbori sa "worst case" situacijom u kojoj inteligentni maliciozni napadač može da probere i simultano obori bilo koja dva čvora na svakih pet minuta.

3.3 Raspored podataka u sistemu

Datoteke se nalaze na onom čvoru na kom su napravljene. I mora da postoji negde kopija te datoteke. U suštini, neki back up. Kod implementacije Chord-a mogu da ostanu gde bi bilo po Chord-u.

3.4 Opšti nefunkcionalni zahtevi

Sistem mora da funkcioniše na pravoj mreži, gde svaka poruka ima proizvoljno kašnjenje i kanali nisu FIFO. Ako se sistem testira na jednoj mašini, neophodno je uvesti veštačka kašnjenja pri slanju poruka, radi realističnog testiranja.

Sva komunikacija mora da bude eksplicitno definisana i dokumentovana. Ako čvoru stigne poruka koja nije po protokolu, treba je odbaciti i ignorisati. Dokumentacija protokola minimalno treba da sadrži sve što je neophodno da se napiše novi servent koji će da učestvuje u radu sistema. Tipično, to je spisak svih poruka koje postoje u sistemu i njihov format – redosled vrednosti koje se prosleđuju, njihov tip i njihovo značenje. Ako postoji neki specifičan redosled slanja poruka, onda navesti i to.