Uvod

Za ovaj projekat se koristi chord sistem gde su nodovi ubaceni u sistem i svaki node je povezan sa jednim successor-om i jednim predecessor-om i samo sa njima komuniciraju u vidu Messages, ako zelim da komuniciram sa nekim node-om koji nije komsija poruku saljemo preko node-ova dok ne dodjemo do zeljenog node-a. Koristimo komande da zahtevamo odredjene stvari od node-ova (komande ce biti objasnjene u sledecem delu). Ovaj projekat koristi algoritam ping/pong da detektuje otkaze ili kvar cvora i izbegevamo gubitke podataka tako sto kopiju podataka jednog cvora cuvamo jos na 2 razlicita cvora. Da ne bi imali probleme u konkuretnosti kao sto su npr race condition koristimo mutex algoritam SuzukiKasami da bi izbegli to (detaljno ce biti objasnjeno u mutexu).

Komande

- add_friend [adresa:port] - Dodaje unutar cvora u kojoj pisemo komandu adresu port koji taj cvor treba smatrati svoji prijateljem
- add_file [path] [private/public] - Dodaje file unutar cvora i biramo da li ce taj fajl biti public ili private
- view_files [adresa:port]- Cvor kome pisemo komandu trazi od drugog cvora fajlove koji taj cvor ima. Prijatelj moze videti i public i private fajlove dok ostali mogu samo public.
- remove_file [filename] - Brisemo fajl iz cvora koji pisemo komandu
- stop - Gasimo cvorove

Messages

Message sadrzi vrednost kao sto su: senderPort, recieverPort, susPort (port za koji sumnjamo da je otkazao ili je otkazao), file koji cuvamo i prosledjujemo, viewPort (port koji zelimo da vidimo njegove podatke), type (Tip poruke), isItDead (specificno za CheckingMessage koji potvrdjuje da je port mrtav). U zavisnosti koji je tip poruke koristimo razlicite promenjive.
Vrste poruka:

- ViewMessage - Poruka koja trazi fajlove od zeljenog cvora (Ako se taj cvor ugasi uzima od onog cvora koji ima kopiju). Prolazi kroz ceo chord uzme podatke i vrati se na kraju u cvor koje je zapoceo poruku.

- UpdateMessage-Poruka koja salje kopiju podatka koji je jedan cvor. Kada cvor ubaci u sebe vrednost salje UpdateMessage i naredna 2 succesora dobiju kopiju tog podatka.

- PutFile- Poruka koja ubacuje podatak u cvor. Prolazi kroz ceo chord i trazi odgovarajuci chordId za napravljeni kljuc od tog fajla. Kljuc se pravi preko chordHash metode tako sto ubacimo string putanje od tog fajla. U koji node se stvalja funkcionise po principu chord sistema

- Ping/PongMessage- Poruke namenjene za proveru stanje cvorova. Cvor i njegov succesor razmenjuju ping/pong poruke i u zavisnosti koliko vremena prodje od pong odgovora (Mi odredjujemo vreme pomocu WEAK_TIMEOUT i STRONG_TIMEOUT) ako previse kasni ili ne posalje uopste poruku automatski formiramo ImpostorMessage i saljemo sledecem succesoru koji on nastavlja da prosledjuje i trazimo od svih cvorova da obrisu mrtav cvor iz njihove memorije. Ukoliko je samo WEAK_TIMEOUT saljemo CheckingMessage prvo da proverimo da li je sve u redu sa node-om koji razmenjujemo Ping/Pong.

- ImpostorMessage - Poruka koja nastave kada detektujemo mrtav node. Poruka prolazi kroz ceo cvor i trazi od primaoca da obrisemo mrtav cvor iz njegove memorije i da ne pokusava da interaktuje vise sa njim.

- CreateCopyMessage - Poruka koja se aktivira kada cvor dobije novi podatak. Node koji dobije novi podatak salje kopiju njegovom succesoru i taj succesor njegovom da cuvaju u njihovom sistemu u slucaju pada cvorova i time smo obezbedili otpornost na otkaze.

- Welcome/Sorry/AskGet- identican ko i pre

- CheckingMessage- Poruka koji salje predecessor succesoru njegovog succesora da proveri da li je cvor aktivan. Poruka se aktivira ako je proslo vise vremena od WEAK_TIMEOUT za slanje pong-a. Kada cvor prihvati poruku proverava da li je njegov precessor poslao skoro PING (zavisi od WEAK_TIMEOUT-a) ako nije vraca poruku i kaze cvoru koji je prvobitno poslao da posalje ImpostorMessage.

- TokenMessage- Poruke koja se koristi sa radom mutexa, bice objasnjeno detaljnije u mutexu.

Detektovanje otkaza

Za detektovanje otkaza smo koristili Ping/Pong algoritam. Cvor salje Ping njegovom succesoru i predecessor Pong kada primi Ping od njega. U zavisnosti koliko vremena prodje od odgovora radimo 3 stvari i 4 stvar ako ne dobijemo uopste odgovor.

1. Pong se posalje na vremene i u tom slucaju nastavljamo normalno sa radom
2. Vreme koje je trebalo da se pong posalje je duze od WeakTimeout-a- u tom slucaju saljemo CheckingMessage njegovom succesoru da proveri da njegovod predecessora. Proveri koliko je vremena proslo od kada je prihvatio ping od njega i ako je proslo odredjen broj vremena (WEAK_TIMEOUT) onda salje CheckingMessage i kaze da moze da se pokrene brisanje node-a iz sistema.
3. Vreme koje je trebalo da se pong posalje je duze od StrongTimeout-a- u tom slucaju automatski saljemo 2 succesoru ImpostorMessage i trazimo da ugasi mrtav cvor i da nastavim poruku.
4. Ne dobije nikakvu poruku posle odredjenog vremena (StrongTimeout) - u tom slucaju je isti kao slucaj 3.

Otpornost na otkaze

U slucaju ako imamo otkaz imamo kopiju podataka tog mrtvog cvora u njegova 2 succesora posto saljemo u UpdateMessage svaki put kada dobijemo novi podatak. Pa ako ViewMessage poruka ne dobije podatak iz zeljenog cvora pokupice njegovu kopiju u neka od 2 naredna cvora.

Mutex

Za mutex smo koristili SuzukiKasami algoritam. Za komunikaciju smo koristili TokenMessage. Node posalje sledecem nodu poruku i prodje kroz ceo chord sistem i vrati se na kraju na u pocetni node kada zavrsi.