**Projekat A6**

**„Mrežni protokol“**

**Dokumentacija**

Radili:

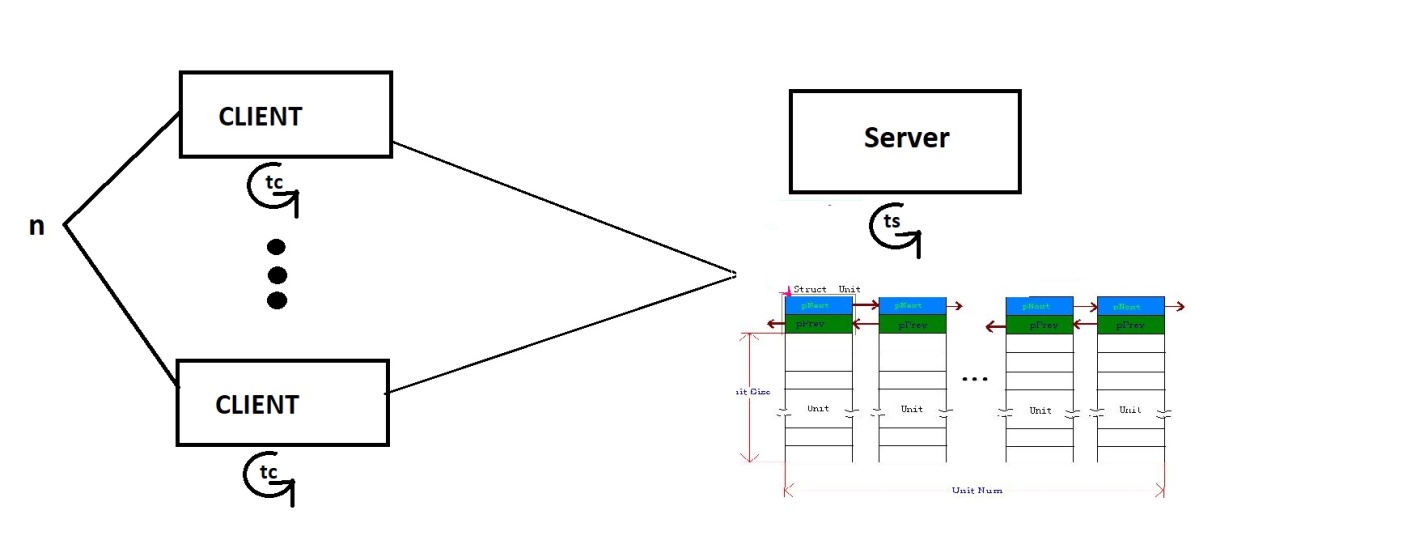
Vladimir Sapundžić pr6-2017

Darie Čolak pr46-2017

***1.Uvod***

Tema Projekta A6 jeste izbegavanje mrežnog zagušenja sa pouzdanim prenosom podataka oslanjajući se na UDP protokol. Na aplikativnom nivou zahteva se implementacija TCP algoritama (Slow-start i Tahoe algoritam). Ovi algoritmi služe za kontrolu prenosa podataka kako bi se izbeglo mrežno zagušenje.

***2.Dizajn***



**Slika 1.Dizajn**

***2.1 Client***

Pre slanja, poruka se deli u segmetne definisane dužine. Segmenti se šalju koristeći algoritam klizajućeg prozora. U implementaciji našeg koda jedan segmet je veličine 4 bajta (može se menjati). Jedan paket se sastoji od jednog segmenta i dodatnih podataka za kontrolu slanja. Client-ska strana pošalje N paketa, sa svojim rednim brojevima od 0 do N-1. Sledeći paket se šalje tek kada je primljena potvrda od strane servera da je paket sa rednim brojem N primljen. Ukoliko ne stigne potvrda od Servera (ACK), radi se retransmisija paketa na Client-u. Upotrebnom clumsly tool-a testirano je da li je implementacija rentrasmisije paketa uspesno izvršena.

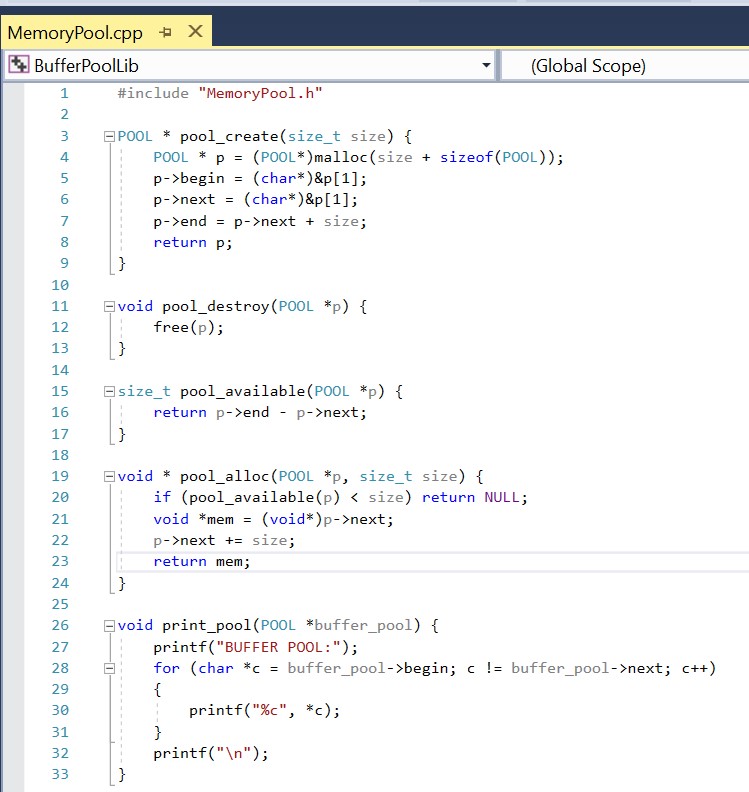
Pored toga, prilikom slanja implementirana je funkcionalnost za izbegavanje mrežnog zagušenja. Širina prozora na početku zadatka postavljena je na 10. Kada se dostigne vrednost parametra X, polovina vrednosti prozora dodeljuje se SSTresh-u , a veličina prozora se vraća na inicijalnu vrednost. Kada se širina prozora izjednači sa SSTresh-om, povećava po formuli SSTresh+SSTresh/windowSize.

***2.2 Server***

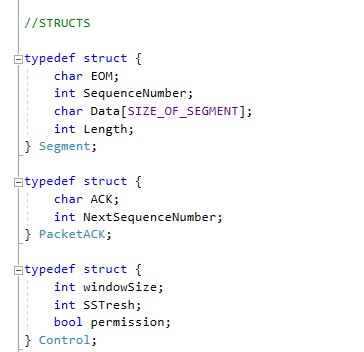
Sa serverske strane se implementira neblokirajući režim. Kada server primi segment smešta ga u buffer pool (memory pool) i šalje potvrdu Client-u.

***3.Strukture podataka***

Prilikom slanja poruka karakteri su se unosili sa tastature. Karatkeri su se smeštali u niz (buffer), koji dalje ide na obradu kako bi se kreirali segmenti. Za potrebe buffer pool-a implementirana je lista (Slika 2). Prilikom kreiranja zauzeta je memorija od 10000 bajtova i postavljeni su pokazivači na početak, kraj iskorišćene memorije i kraj zauzete memorije, radi lakšeg upisa segmenta u buffer pool. Koriste se strukture radi kreiranja segmenta, ACK-a i primene algoritma protiv zagušenja mreže (Slika 3).



**Slika 2.Lista buffer pool**

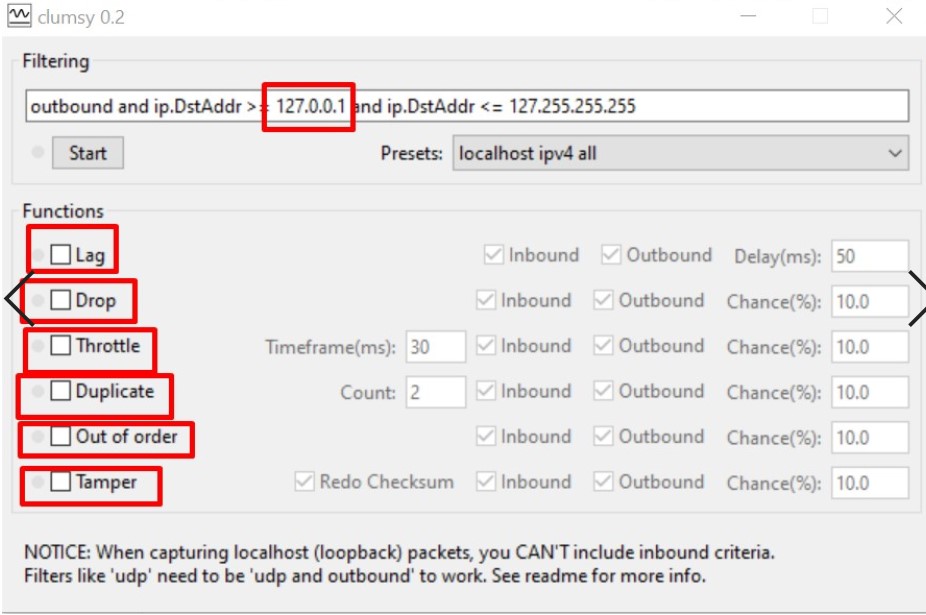


**Slika 3. Struct**

***4.Rezultati testiranja***

Rezultati testiranja da li je memorija koja je zauzeta za potrebe buffer pool-a oslobođena su priloženi u vidu slika u folderu „Snapshots”. Slike su napravljene nakon zauzimanja memorije (prva.jpg i druga.jpg), pre prijema segmenta (treca.jpg i cetvrta.jpg), nakon prijema poruke (peta.jpg), nakon izlaska iz aplikacije ( sesta.jpg i sedma.jpg).

Rezultati testiranja clumsy tool-om nalaze se u folderu „ClumsyTest“(slike).



**Slika 4. Clumsy**

Čekiranjem jednog od kvadratića korisnik omogućuje pogoršanje stanja u mreži. „Lag“ - zadržavanje paketa do 3 sekunde (3000ms) u mreži. „Drop“ - gubitak paketa u mreži. „Throttle“ - blokiranje saobraćaja za dati vremenski okvir (Timeframe), zatim slanje u seriji. „Out of order“ - preusmeravanje redosleda paketa. „Tamper“ - dodavanje slučajnog sadržaja u paket.

***5.Zaključak***

Iz navedenih slika u “Snapshots” folderu, možemo zaključiti da se memorija koja se zauzme nakon inicijalizacije buffer pool-a i oslobodi nakon gašenja aplikacije. Testiranjem našeg koda kroz aplikaciju clumsy tool, govori nam o tome da i pored smetnji u mreži naši paketi će stići i ACK će biti uspešno vraćen ka Client-u.