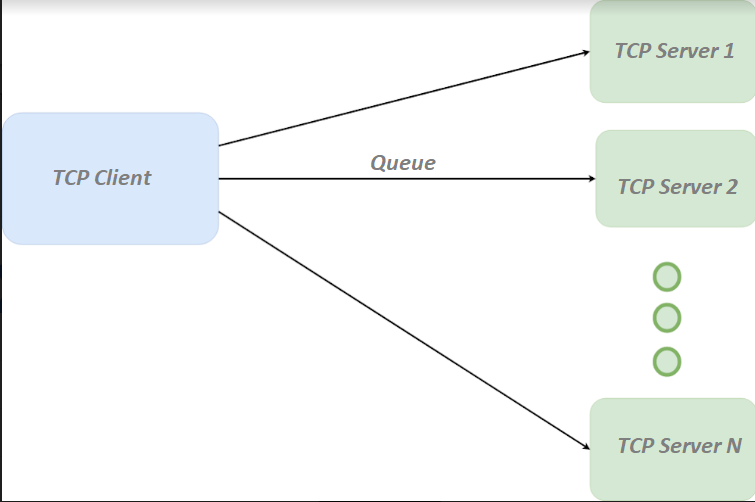
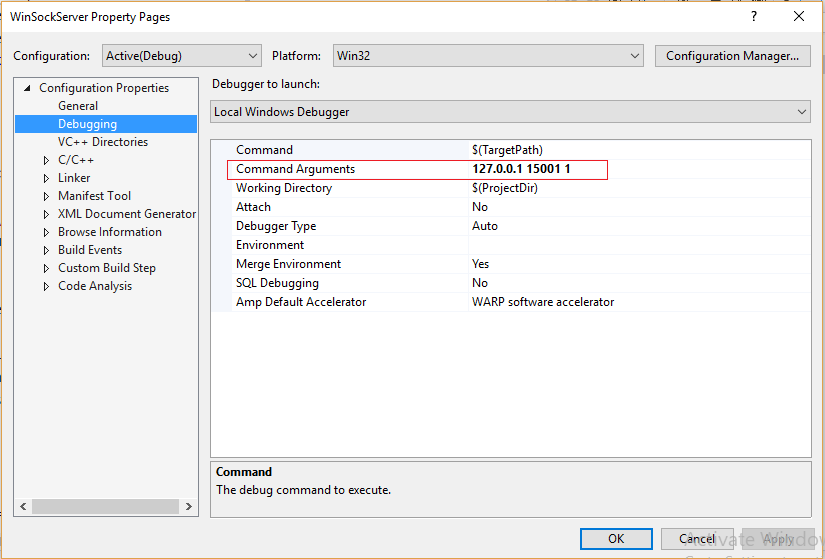
***Arhitektura servisa***



***Uputstvo:***

Unutar *Solution Exlorer* odabrati projekat pod nazivom *WinSockServer->Properties.*

Sa leve strane nalazi se *Configuration Properties* meni, označiti drugu opciju *Debugging.* U *Command Arguments* polju uneti tekst kao sa slike ispod.

Zatim kliknuti *OK* i startovati instancu programa. Pritiskom na dugme *Debug.*

Sada smo pokrenuli *TCP klijenta* čiji je zadatak da pošalje poruku nekom od servera (koje ćemo pokrenuti u sledećem koraku) tako što će uneti Sadržaj poruke i ID *TCP Servera*, ali tek nakon pokretanja servera. Broj servera je ograničen na 10.

**Korak 2.**

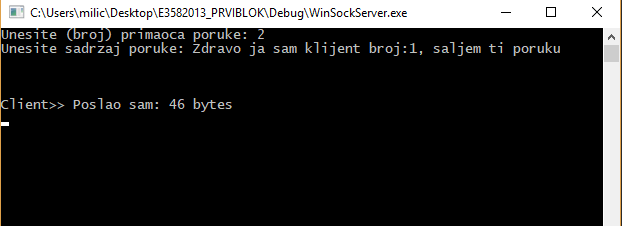
Kada smo pokrenuli klijenta, sledeći korak je da pokrenemo prvu instancu servera. Desni klik na *WinSockServer,* zatim biramo opciju Properties, i ponavljamo prvi korak, samo što sada u polju *Command Arguments* unosimo sledeće: **127.0.0.x 1500x x (x je proizvoljan celobrojni jednocifren broj, i označava ID servera, gde je x≠1).**

Prilikom kreiranja svakog narednog servera ponavljamo **korak 2**, samo umesto x unosimo y (takođe proizvoljan celobrojan jednocifren broj, gde je y≠x ^ y≠1 ). Server pokrećemo tako što desnim klikom na *WinSocketServer* odaberemo *Debug->Start new instance.*

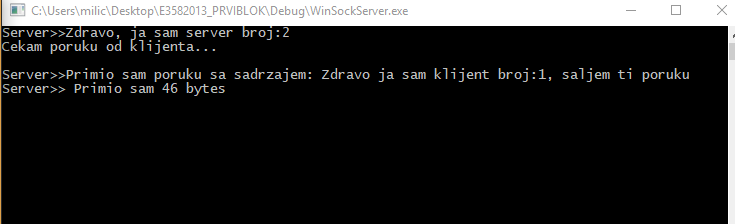
Kada smo pokrenuli najmanje jedan server, možemo mu poslati poruku.

Klijent inicira komunikaciju i šalje serveru poruku, server prima poruku.

***TCP Client>***



***TCP Server>***



Klijent i Server komuniciraju putem ***socketa*** koji rade u ***neblokirajućem režimu.*** Podaci se šalju putem memorijskog bafera implementiranog preko ***queue*** strukture.

Ova struktura sadrži *SOCKET* (IP adresu i port), *veličinu memorijskog bafera* izraženog u bajtima i *sadržaj memorijskog bafera.*

Aplikacija koristi i već implementiranu strukturu **queue**, na predajnoj (klijentskoj) strani se izvršava **push** naredba za stavljanje sadržaja u red (queue), a na prijemnoj (serverskoj) **front** naredba, sa preuzimanje sadržaja iz reda.

TCP protokol garantuje isporuku poruke, i da će poruka stići u redosledu u kom je poslata, a neblokirajući režim rada soketa obezbeđuje da glavna nit koja čeka podatke na soketu ne bude blokirana dok podaci ne stignu, i time dovede do zagušenja podataka. U blokirajućem režimu rada soketa, paketi se ne šalju dok se ne primi potvrda za poslati paket, ovaj pristup nazvan je stop-and-wait algoritam, i ima za posledicu nisko iskorišćenje mrežne infrastrukture. Mehanizam koji ovaj problem rešava se naziva princip klizajućeg prozora. TCP obezbeđuje i kontrolu zagušenja i kontrolu toka.